LEARNING STATISTICS IN THE COMPUTER LAB

by

Brian Knaeble

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mathematics

The University of Utah

August 2012

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Brian Knaeble**

has been approved by the following supervisory committee members:

Lajos Horvath	, Chair	**4/10/2012** Date Approved
Andrejs Treibergs	, Member	**4/10/2012** Date Approved
Nelson Beebe	, Member	**4/10/2012** Date Approved
Hugo Rossi	, Member	**4/10/2012** Date Approved
Donna Ziegenfuss	, Member	**4/10/2012** Date Approved

and by **Peter Trapa** , Chair of the Department of **Mathematics**

and by Charles A. Wight, Dean of The Graduate School.

ABSTRACT

The purpose of this dissertation was to create a textbook that supplements traditional statistics curriculum. Emphasis was placed on statistical computing. Multiple computing environments were used to demonstrate essential skills for the applied statistican. Mathematical theory was developed in order to make the text self contained. A technique was developed to replace a faulty command for computing moment generating functions within Mathematica.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This text has grown out of the need to add computing curriculum to an upper division course in mathematical statistics at the University of Utah. It has been designed to supplement theoretical treatment of the subject at the level of Bain and Engelhardt's *Introduction to Probability and Mathematical Statistics*. Readers should have knowledge of introductory Real Analysis and Probability Theory, and some exposure to Linear Algebra. A background in computer programming is not required.

The text gently introduces Mathematica, Maple, SPSS, SAS and R. Students develop basic computing literacy and improve their understanding of statistics.

1.1 Many languages

Rather than focus on a single computing language we have decided to introduce many different environments. This is because different software can sometimes give different answers to the same problem. The following example serves to explain how even the best of programs can contain errors. Mathematica has recently [1] advertised:

> What's new in Mathematica 8? Mathematica 8 introduces free-form linguistic input — a whole new way to compute. Enter plain English; get immediate results — no syntax required. It's a new entry point into the complete Mathematica workflow, now upgraded with 500 additional functions and 7 application areas, including the world's most advanced statistics capability and state-of-the-art image processing.

This has led to confusion because the "world's most advanced statistics capability" has been found to compute moment generating functions incorrectly in some cases. We show an example of this in Chapter 1. For now the lesson is simply that mistakes can happen, even with glitzy computer packages. In order to sort through the resulting confusion one should be prepared to work with multiple computing environments.

1.2 Educational component

In addition to computer programming, this text also teaches statistics. The five previously-mentioned computing environments are introduced as tools to be used when solving practical

questions that arise in the social sciences, engineering and the physical sciences. Real data are used through the book, and after learning to carry out appropriate statistical analyses, students can expect to find themselves comfortable with coding and also more receptive to the rigors of theoretical statistics—the mathematical definition of a distribution makes much more sense after seeing multiple stripcharts plotted in R.

We have chosen data that illustrate key theoretical concepts of statistics. For example, when comparing population means, the data are such that it is not clear whether we should accept the standard assumption of equal variances. Upon rejecting the assumption we are forced to deal with more complicated mathematics, and in general we welcome such inconveniences within this text. Prepare to get your hands dirty as you learn to do statistics with computers.

1.3 For the instructor

This supplemental text is meant to be used at the level of *Introduction to Probability and Mathematical Statistics* by Bain and Engelhardt. It can also supplement courses that use books such as the following: *Probability and Statistics* by Morris H. DeGroot, *Advanced Statistics from an Elementary Point of View* by Michael J. Panik, *Mathematical Statistics* by Miller and Miller, and other similar texts.

This text, *Learning Statistics in the Computer Lab*, is meant to supplement a year-long course in statistics at the advanced undergraduate level or the beginning graduate level. As such, instructors can plan for students to complete about 25 modules. A module typically consists of reading a section of a chapter and completing the accompanying exercises. However, some sections are introductory and do not contain exercises. These should not be treated as modules. Also, some sections are lengthy and can be broken into two modules. Modules can be done in any order. However, because each chapter is devoted to a particular computing environment, it is recommended to complete all the sections from a given chapter at once. Table 1.1 shows a sample schedule for a year's worth of modules. Instructors can simply specify a weekly computing exercise as part of their traditional homework assignments, and then students can read what they may of this text in order to complete the assigned exercise. Certain exercises require datasets that can be downloaded at `http://www.math.utah.edu/~knaeble/Datasets.html`.

Table 1.1. A sample schedule

Semester.Week	Theoretical Topic	Computing Topic	Exercise
1.1	Sums of random variables	MGF (discrete)	2.1
1.2	Transformation methods	MGF (continuous)	2.2
1.3	Sequences	MGF (using the CDF)	2.3
1.4	Central Limit Theorem	MGF (piecewise)	2.4
1.5	Any	Introduction to R	3.2
1.6	Any	Computation in R	3.1
1.7	Common distributions	Accuracy of simulations	3.3
1.8	Limit Theorems	Sufficient sample size?	3.4
1.9	Any	Writing programs in R	3.5
1.10	Order statistics	Display of R graphics	3.6
1.11	Extreme order stats	Asymptotic distributions	3.7
1.12	Cauchy distribution	Extreme order statistics	3.8
1.13	Introduction to MLE	Optimization	4.1
1.14	MLE continued	Numerics for MLE	4.2
1.15	MLE of a vector	Vectors of MLEs	4.3
2.1	Introduction to CIs	Importing data	5.1
2.2	CIs continued	Basic programming	5.2
2.3	Pivotal quantities	CI for mean	5.3
2.4	General CI mathod	CI for proportion	5.4
2.5	T and F distributions	Introduction to SPSS	6.1
2.6	Hypothesis testing	Continued introduction	6.2
2.7	Test assumptions	Two sample means	6.3
2.8	ANOVA	Three sample means	6.4
2.9	Introduction to regression	Plotting in R	7.1
2.10	Simple linear regression	Checking assumptions	7.2
2.11	Continuation	Continuation	7.3
2.12	Lurking variables	Interpreting printout	7.4
2.13	General linear model	Fitting a model	7.5
2.14	Fitted coefficients	Model selection	7.6
2.15	Goodness of fit	R skills	7.7
2.16	Bootstrapping	Using ‘apply’	7.8

CHAPTER 2

MOMENT GENERATING FUNCTIONS WITH MATHEMATICA

In this chapter we learn how to compute moment generating functions with Mathematica. In general, Mathematica can effectively compute moment generating functions. However, in certain instances a particular procedure will result in an incorrect answer. We illustrate this pitfall in Section 2.2.3.

2.1 Useful formulas

Moment generating functions are related to Laplace transforms, the latter of which can often be obtained via computational software. In this preliminary section we lay out the mathematics that relates moment generating functions to Laplace transforms.

Recall that the moment generating function, $M(t)$, of a real-valued, random variable X is defined as

$$M_X(t) = E(e^{tX}).$$

The cumulative distribution function, $F(x) = P[X \leq x]$, can then be employed to express this expectation as an integral:

$$E(e^{tX}) = \int_{-\infty}^{\infty} e^{tx} dF(x).$$

When X is discrete, taking the values, $\{x_i\}_{i=1}^{\infty}$, each with probability p_i, where $\sum_{i=1}^{\infty} p_i = 1$, then this integral can be written as a sum:

$$M_X(t) = E(e^{tX}) = \int_{-\infty}^{\infty} e^{tx} dF(x) = \sum_{i=1}^{\infty} e^{tx_i} p_i. \qquad \text{(discrete case)}$$

When X has a probability density function $f(x)$ then the expectation can be expressed as

$$E(e^{tX}) = \int_{-\infty}^{\infty} e^{tx} dF(x) = \int_{-\infty}^{\infty} e^{tx} f(x) dx.$$

The form of this integral is reminiscent of the two-sided Laplace transform of the function $f(x)$. We will be interested in the one-sided Laplace transform, because it is readily implemented within Mathematica.

Definition 2.1. *(Laplace transform) Given a function $h(x)$ we can define the (one sided) Laplace Transform $\mathscr{L}\{f(x)\}$ to be the following function of t:*

$$\mathscr{L}\{f(x)\}(t) = \int_0^\infty e^{-xt} h(x)\,dx.$$

Pay attention to the minus sign in the exponent, and the bounds on the integral. They distinguish the Laplace-transform integral from the moment-generating-function integral in the case where the random variable has a probability density function. If we further assume that the random variable is non-negative valued then we have the following.

$$M_X(t) = E(e^{tX}) = \int_{-\infty}^\infty e^{tx}\,dF(x) = \int_{-\infty}^\infty e^{tx} f(x)\,dx = \int_0^\infty e^{tx} f(x)\,dx = \mathscr{L}\{f(x)\}(-t).$$

Even if we allow negative values, as long as the random variable has a probability density function, we can proceed by splitting the domain of integration and then manipulating the expressions until we get (roughly) two Laplace transforms.

$$\begin{aligned} M_X(t) = E(e^{tX}) = \int_{-\infty}^\infty e^{tx}\,dF(x) &= \int_{-\infty}^\infty e^{tx} f(x)\,dx && \text{(continuous case)} \\ &= \int_{-\infty}^0 e^{tx} f(x)\,dx + \int_0^\infty e^{tx} f(x)\,dx \\ &= \int_0^\infty e^{-tx} f(-x)\,dx + \int_0^\infty e^{tx} f(x)\,dx \\ &= \mathscr{L}\{f(-x)\}(t) + \mathscr{L}\{f(x)\}(-t). \end{aligned}$$

For random variables that are neither discrete nor continuous (for such an example see [6], Chapter 6, Section 3) we can still use the Laplace transform. In fact, the following formula can be derived (see Section 2.3):

$$M_X(t) = -t\mathscr{L}\{F(-x)\}(t) + t\mathscr{L}\{1 - F(x)\}(-t) + 1. \qquad \text{(general formula)}$$

This result shows the general utility of the Laplace transform when computing moment generating functions. It is recommended to use this most-complicated formula when $F(x)$ is more readily defined than $f(x)$. When $f(x)$ is readily defined or when the random variable is discrete, the previous formulas can still be used. Table 2.1 summarizes what we have developed thus far.

2.2 Computing with Mathematica

Here we will examine how a computer algebra system, such as Mathematica, can help with the computation of moment generating functions. For further reading on Mathematica see [2].

Table 2.1. Formulas for moment generating functions $M_X(t)$

Conditions on X	$M_X(t)$
• real-valued • discrete	$\sum_{i=1}^{\infty} e^{tx_i} p_i$
• real-valued • absolutely continuous • non-negative valued	$\mathscr{L}\left\{f(x)\right\}(-t)$
• real-valued • absolutely continuous	$\mathscr{L}\left\{f(-x)\right\}(t) + \mathscr{L}\left\{f(x)\right\}(-t)$
• real-valued	$-t\left(\mathscr{L}\left\{F(-x)\right\}(t) + t\mathscr{L}\left\{1-F(x)\right\}(-t)\right) + 1$

2.2.1 Discrete random variables

We begin with a generic discrete random variable X, taking countably many values $\{x_i\}_{i=1}^{\infty}$, each with probability p_i where $\sum_{i=1}^{\infty} p_i = 1$. As shown in Section 2.1 the moment generating function $M_X(t)$ is given by $\sum_{i=1}^{\infty} e^{tx_i} p_i$. So if the points x_i and the probabilities p_i are given by formulas, we can try the following command:

```
Sum[Exp[tx_i]p_i,{i,1,Infinity}]
```

Slight modifications may prove beneficial. For example, if $X \sim POI(\mu)$ then it makes sense to shift the indexing set to include zeros, so that $x_i = i$ and $p_i = e^{-\mu}\mu^i/i!$ for $i = 0,1,2,...$, and the moment generating function can be obtained with the following command:

```
Sum[Exp[t i]Exp[-\[Mu]]\[Mu]^i/(i!),{i,0,Infinity}]
```

If using Mathematica-8.0.0 or higher there are many built in statistical tools. In fact, for common distributions such as the Poisson distribution, POI(μ), the moment generating function can be computed directly, using the following command:

```
MomentGeneratingFunction[PoissonDistribution[\[Mu]],t]
```

The beginner should take note that within Mathematica 'enter' merely moves the cursor to the next line. 'shift-enter' is required to execute a command. To see exactly what Mathematica displays after entering the previously mentioned command see the supplementary material.

2.2.2 Working with a probability density function

When a probability density function exists it can be used to define the distribution. Once defined, the distribution's moment generating function can be obtained by using 'MomentGeneratingFunction'. Alternatively, one can apply the Laplace transform to the density directly. We illustrate both of these techniques for exponential distributions and just the latter Laplace-transform technique for the double exponential distribution. As we will see in the following section, the Laplace-transform technique can be more reliable.

The exponential distributions have probability density functions of the form

$$f_X(x) = \frac{1}{\theta} e^{-x/\theta} \mathrm{I}_{\{x>0\}}.$$

We will use this family of functions to define the exponential family of distributions within Mathematica. First we define the family of functions:

```
f:=(1/\[Theta] Exp[-x/\[Theta]])Boole[0<x]
```

Then we define the family of distributions, labeled with '`exp`':

```
exp[\[Theta]]=ProbabilityDistribution[f,{x,0,\[Theta]}]
```

'MomentGeneratingFunction' then computes the moment generating function. To view the whole sequence of commands and the result as displayed within Mathematica see the supplementary material.

Because exponential distributions are absolutely continuous and defined on positive real numbers, their moment generating functions can be obtained by computing a single Laplace transform, evaluated at $-t$, as outlined in Section 2.1. In Mathematica this involves (again) defining the function.

```
f:=(1/\[Theta] Exp[-x/\[Theta]])Boole[0<x]
```

Then the Laplace transform can be computed, via

```
LaplaceTransform[f,x,-t]
```

The result can be simplified by typing '`Simplify[%]`'. For a picture of the resulting Mathematica output see the supplmentary material.

Next we focus on the double-exponential family of distributions. A random variable with a double-exponential distribution has a probability density function of the form

$$f(x) = \frac{1}{2\theta} e^{-|x-\eta|/\theta}.$$

Since the double-exponential distribution is defined on the entire real line, a single Laplace transform will not suffice for computing the moment generating function. However, since the double-exponential distribution is absolutely continuous, we can use the 'continuous case' formula from Section 2.1 on page 5, which expresses the moment generating function as $\mathscr{L}\{f(-x)\}(t) + \mathscr{L}\{f(x)\}(-t)$. In order to implement this within Mathematica we define both $f(x)$ and $f(-x)$ which we will label with g.

```
f:=1/(2\[Theta]) Exp[-Abs[x-\[Eta]]/\[Theta]]
g:=1/(2\[Theta]) Exp[-Abs[-x-\[Eta]]/\[Theta]]
```

The Laplace transform calculations can be carried out as long as we specify that η is a real number.

```
Assuming[Element[\[Eta],Reals],LaplaceTransform[g,x,t]+LaplaceTransform[f,x,-t]]
```

To view the entire ouput see the supplementary material

While the calculation could have been done by hand, our computational techniques work well in general. As an example consider a random variable with probability density function

$$f(x) = \frac{5}{2\theta^5} x^4 \mathrm{I}_{\{-\theta<x<\theta\}}.$$

Its moment generating function can be computed through repeated integrations by parts, but this approach is tedious and prone to errors. We have at our disposal two separate techniques that can both be applied quickly. To view Mathematica commands that demonstrate the equivalence of both techniques see the supplementary material.

2.2.3 Working with a distribution function

In this section we explore the possibility of defining a distribution within Mathematica by specifying not the probability density function, but rather the cumulative distribution function.

This generalizes the approach of the previous section so that we are equipped to handle any random variable, not just discrete or absolutely continuous random variables. We first illustrate the approach using a relatively simple random variable with distribution function given by

$$F(x) = \begin{cases} \dfrac{1}{4}x + \dfrac{1}{2} & -1 \le x \le 1 \\ \dfrac{\arctan(x)}{\pi} + \dfrac{1}{2} & \text{otherwise.} \end{cases} \tag{2.1}$$

Figure 2.1 displays the graph of $F(x)$, which can be defined in Mathematica as follows:

```
F:=Piecewise[{{x/4+1/2,-1<=x<=1}},ArcTan[x]/\[Pi]+1/2]
```

We then proceed to define the distribution, labeled as `'dist2'`, by specifying its cumulative distribution function (F) and also its range.

```
dist2=ProbabilityDistribution[{"CDF",F},{x,-Infinity,Infinity}]
```

Then we can compute the moment generating function. To view the complete set of commands and the resulting output see the supplementary material, where results obtained via the Laplace transform method are displayed as well. Presumably these results should match , but there is a concern that runs deeper than simply comparing the answers to see if we entered the code correctly. As the following simple example illustrates, the 'MomentGeneratingFunction' command

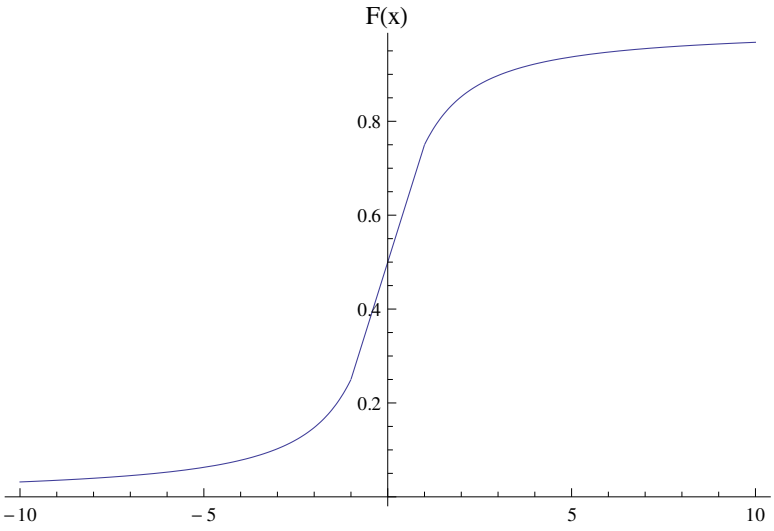


Figure 2.1. The graph of $F(x)$ from (2.1)

should not be applied to a distribution containing any point masses! Figure 2.2 displays the code (along with a plot) that we might mistakenly use to compute $M_X(t)$ with X defined via

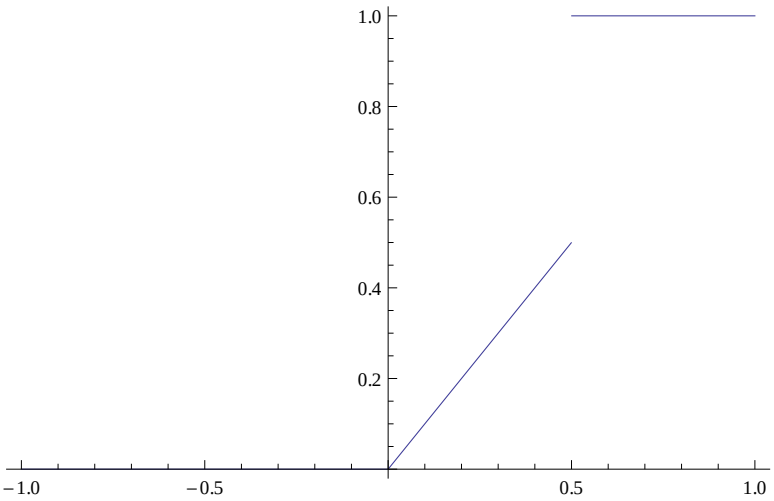
$$F(x) = \begin{cases} 0 & x \leq 0 \\ x & 0 < x < 1/2 \\ 1 & 1/2 \leq x. \end{cases}$$

The result can be seen to be incorrect through a direct computation (see exercises). The lesson here is that one should often check their results against those obtained using alternative methods. In this case we have found the 'MomentGeneratingFunction' command to work fine for absolutely continuous distributions, but it fails to account for point masses. This glitch is currently being worked on and could be fixed in future editions of Mathematica. The suggested fix: try using our general Laplace transform method and package it under the name 'MomentGeneratingFunction'.

As a final example consider a random variable X with cumulative distribution function

```
In[1]:= F := Piecewise[{{0, x ≤ 0}, {x, 0 < x < 1 / 2}}, 1]

In[2]:= Plot[F, {x, -1, 1}]
```

Out[2]=




```
In[3]:= sjdist := ProbabilityDistribution[{"CDF", F}, {x, -Infinity, Infinity}]

In[4]:= MomentGeneratingFunction[sjdist, t]

Out[4]= (-1 + e^(t/2)) / t
```

Figure 2.2. 'MomentGeneratingFunction' fails to account for the point mass

$$F(x) = \begin{cases} 0 & x < -1 \\ \frac{x+1}{4} & -1 \le x < 0 \\ \frac{x^4+1}{2} & 0 \le x \le 1 \\ 1 & 1 < x. \end{cases}$$

Its graph is displayed in Figure 2.3. This distribution has a point mass so we should not use the 'MomentGeneratingFunction' command. Instead we use the general Laplace-transform formula. To view the output see the supplementary material.

Exercise 2.1. *For $X \sim POI(\mu)$ compute $M_X(t)$ using Mathematica's 'Sum' command.*

Exercise 2.2. *Find an absolutely continuous random variable whose moment generating function would be difficult to compute by hand. Use mathematica and both of the above techniques to compute the moment generating function.*

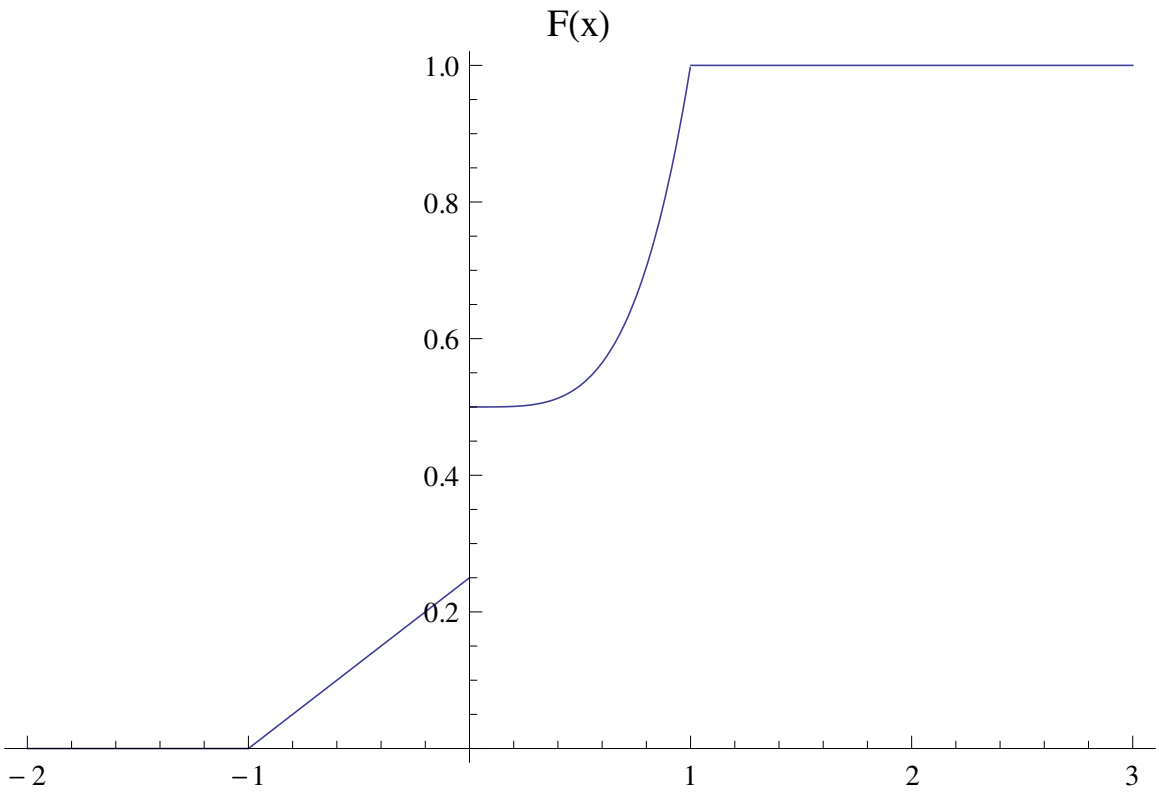


Figure 2.3. A more complicated distribution with a point mass

Exercise 2.3. *With X being distributed according to the distribution function*

$$F(x) = \begin{cases} 0 & x \le 0 \\ x & 0 < x < 1/2 \\ 1 & 1/2 \le x, \end{cases}$$

compute $M_X(t) = E(e^{tX}) = \int_{-\infty}^{\infty} e^{tx} dF(x)$ by hand. Compare with the result obtained via 'MomentGeneratingFunction'. Try using the most general, Laplace-transform formula.

Exercise 2.4. *With X having probability density function*

$$f(x) = \begin{cases} (x+2)/3 & -2 < x < -1 \\ 1/3 & -1 \le x \le 1 \\ (2-x)/3 & 1 < x < 2 \\ 0 & Otherwise \end{cases},$$

use Mathematica to compute $M_X(t)$.

2.3 Derivation of the Laplace-transform formula

In this section we prove the validity of the following formula that expresses any moment generating function in terms of Laplace Transforms:

$$M_X(t) = -t\mathscr{L}\{F(-x)\}(t) + t\mathscr{L}\{1 - F(x)\}(-t) + 1.$$

We begin with a lemma.

Lemma 2.2. *Let X be a random variable with distribution function $F(x)$ such that*

$$M_X(t) = \int_{-\infty}^{\infty} e^{-tx} dF(x) < \infty$$

for some t. We then have

$$\lim_{x \to -\infty} e^{tx} F(x) = 0 \text{ and } \lim_{x \to \infty} e^{tx}(1 - F(x)) = 0.$$

Proof. It will be convenient to introduce the dummy variable y as follows.

$$\begin{aligned} \infty > \int_{[-\infty,\infty]} e^{tx} dF(x) &= \int_{[-\infty,0)} e^{tx} dF(x) + \int_{[0,\infty]} e^{tx} dF(x) \\ &= \int_{[-\infty,0)} e^{tx} dF(x) - \int_{[0,\infty]} e^{tx} d(1 - F(x)) \\ &= \int_{[-\infty,0)} e^{ty} dF(y) - \int_{[0,\infty]} e^{ty} d(1 - F(y)). \end{aligned} \tag{2.2}$$

Depending on the sign of t one of the integrals from (2.2) will be finite due to integrating a bounded function over a finite measure; the other is then seen to be finite as well due to the inequality. Thus, both integrals from line 2.2 must be finite. Thus,

$$\infty > \int_{[-\infty,0)} e^{ty} dF(y) = \sum_{i=0}^{\infty} \int_{(-[i+1),-i)} e^{ty} dF(y) \implies \lim_{x\to-\infty} \int_{-\infty}^{x} e^{ty} d(F(y)) = 0,$$

since each summand is positive. Also,

$$-\infty < \int_{[0,\infty]} e^{ty} d(1-F(y)) = \sum_{i=0}^{\infty} \int_{[i,i+1)} e^{ty} d(1-F(y)) \implies \lim_{x\to\infty} \int_{x}^{\infty} e^{ty} d(1-F(y)) = 0,$$

since each summand is negative.

Hence, when $t < 0$

$$\begin{aligned}
\lim_{x\to-\infty} e^{tx} F(x) &= \lim_{x\to-\infty} e^{tx} \int_{-\infty}^{x} d(F(y)) \\
&= \lim_{x\to-\infty} \int_{-\infty}^{x} e^{tx} d(F(y)) \\
&\leq \lim_{x\to-\infty} \int_{-\infty}^{x} e^{ty} d(F(y)) = 0.
\end{aligned}$$

The inequality is an equality since the integrals are positive valued. When $t \geq 0$, $\lim_{x\to-\infty} e^{tx} F(x) = 0$ trivially.

Similarly, when $t > 0$

$$\begin{aligned}
\lim_{x\to\infty} e^{tx}(1-F(x)) &= \lim_{x\to\infty} (-e^{tx}) \int_{x}^{\infty} d(1-F(y)) \\
&= \lim_{x\to\infty} \int_{x}^{\infty} -e^{tx} d(1-F(y)) \\
&\leq \lim_{x\to\infty} \int_{x}^{\infty} -e^{ty} d(1-F(y)) = 0.
\end{aligned}$$

Again, the inequality is an equality. Finally, when $t \leq 0$ $\lim_{x\to\infty} e^{tx}(1-F(x)) = 0$ trivially. This completes the proof. □

Lemma 2.2 will be employed in the proof of Theorem 2.3 below. A generalized integration-by-parts formula will also be used (Hewitt and Stromberg [3] Theorem 21.67). Their technical

formulation provides more generality than we need, but quickly justifies the following formally-familiar equalities,

$$\int_{[-\infty,0]} e^{tx} dF(x) = e^{tx} F(x)|_{-\infty}^{0} - \int_{[-\infty,0]} F(x)\, t e^{tx} dx$$

and

$$\int_{[0,\infty]} e^{tx} d(1-F(x)) = e^{tx}(1-F(x))|_{0}^{\infty} - \int_{[0,\infty]} (1-F(x))\, t e^{tx} dx$$

which will be used in our upcoming derivation. Upon applying these formulas we will face the need to evaluate $F(x)$ at the boundary of closed intervals. According to the general integration by parts formula, this value will always be the limit as x approaches the boundary from the outside. We are now ready to prove the general formula.

Theorem 2.3. *For any X, and t such that $M_X(t) < \infty$, we also have*

$$M_X(t) = -t\mathscr{L}\{F(-x)\}(t) + t\mathscr{L}\{1-F(x)\}(-t) + 1$$

Proof. Using the integration by parts formula of Hewitt and Stromberg,

$$\begin{aligned}
M_X(t) = E(e^{tX}) &= \int_{[-\infty,\infty]} e^{tx} dF(x) \\
&= \int_{[-\infty,0]} e^{tx} dF(x) + \int_{[0,\infty]} e^{tx} dF(x) - P[X=0] \\
&= e^{tx} F(x)|_{-\infty}^{0} - \int_{[-\infty,0]} F(x)\, t e^{tx} dx - \int_{[0,\infty]} e^{tx} d(1-F(x)) - P[X=0] \\
&= F(0) - \int_{[0,\infty]} F(-x)\, t e^{-tx} dx - e^{tx}(1-F(x))|_{0}^{\infty} \\
&+ \int_{[0,\infty]} (1-F(x))\, t e^{tx} dx - P[X=0] \\
&= F(0) - t\mathscr{L}\{F(-x)\}(t) + 1 - \lim_{x\to 0^-} F(x) \\
&+ t\mathscr{L}\{1-F(x)\}(-t) - P[X=0] \\
&= -t\left(\mathscr{L}\{F(-x)\}(t) + t\mathscr{L}\{1-F(x)\}(-t)\right) + 1
\end{aligned}$$

□

CHAPTER 3

CONDUCTING SIMULATIONS WITH R

R is a programming language and software environment for doing statistics. The peculiar name R is partly a play on the programming language S, partly resulting due to the fact that the two, original authors both had first names beginning with the letter R ([11]). R is freely available for download at http://cran.r-project.org/. The same site is useful for reading more about R. In what follows here and in a later chapter we will focus on the statistics that R can do. In particular, this chapter will demonstrate how R can be used to conduct simulations.

3.1 Basic computation

We first learn how R can act as a fancy calculator. R does operations on real numbers as we will see shortly. This should be viewed as a special case of R doing operations on matrices (a one by one matrix is a real number). Matrices are in turn special cases of arrays (a Matrix is a two-dimensional array) so we could say that R does operations on arrays, but to keep this introduction simple we prefer to state that R does operations on matrices. Thus, if your linear algebra skills are not up to par, now is the time to review. An excellent resource is Gilbert Strang's book (Linear Algebra and its Applications [9]). With a solid understanding of linear algebra it becomes expedient to think of a dataset as a matrix, especially when the data are numeric. The observations are the rows of the matrix and the variables are the columns of the matrix. Keep this in mind as we learn here how to work with matrices using R.

Here we do simple arithmetic on the simplest of matrices, just numbers.

```
> 7+4
> 7-4
> 2*3
> 6/2
```

Multiplication of numbers is done with *, exponentiation with ^

```
> 3^4
> log(2.71)
[1] 0.9969486
> log10(10)
[1] 1
```

'log' by itself is the natural log.

Here we move onto vectors, starting with a short vector of length one. We are defining these objects on the right and labeling them with the symbol to the left. In between we see '< −' thought of as a single symbol. An equals sign works just as well.

```
> x<-5
> y<-seq(1:12)
> z<-c(5,4,8)
> a<-c(y,z)
> a
 [1]  1  2  3  4  5  6  7  8  9 10 11 12  5  4  8
```

'seq' makes a sequence of numbers, in this case 1 through 12. 'c' stands for concatenate. We concatenate 'z' to 'y' resulting in 'a'. We have defined the objects 'x', 'y', 'z', and 'a'. We display an object, such as 'a' by typing the name, 'a', and then hitting enter. We can think of one-dimensional lists of numbers such as these as vectors.

When the numbers are arranged in a two-dimensional array we call it a matrix.

```
> A<-matrix(a,nrow=5)
> A
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8    5
[4,]    4    9    4
[5,]    5   10    8
> B<-matrix(y,nrow=3)
> B
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

The function 'matrix' snakes a vector of numbers, column wise, into the form of a matrix. We simply stated the vector (list) of numbers to be used and specified the resulting shape (number of rows, number of columns) of the resulting matrix. The length of the vector of numbers combined with the number of rows or number of columns is enough to determine the shape.

The syntax is not as tricky as it may appear. The function 'matrix' has many arguments. We specified two. The remaining, unspecified arguments are set to the defaults. When wanting to learn more about a function one can always get help (within R) by typing '?' followed by the function name.

In contrast to the curved brackets that are commonly used with functions, we use square brackets to pick out parts of a matrix.

```
> A[3,2]
[1] 8
> v_3<-B[,3]
> v_3
[1] 7 8 9
> A[c(1,5),c(1,3)]
     [,1] [,2]
[1,]    1   11
[2,]    5    8
```

Remember, operations such as these correspond to cutting to certain parts of your dataset that happen to be of interest.

Here are some standard mathematical operations: 't' for transpose, 'solve' for inverse, 'eigen' as a function that computes eigen values and vectors, 'diag' as a function that isolates the diagonal, and 'sum' which sums the entries of a vector.

```
> C=A[c(1,5),c(1,3)]
> t(C)
     [,1] [,2]
[1,]    1    5
[2,]   11    8
> solve(C)
           [,1]        [,2]
[1,] -0.1702128  0.23404255
[2,]  0.1063830 -0.02127660
> eigen(C)
values
[1] 12.70061 -3.70061

vectors
           [,1]       [,2]
[1,] -0.6849572 -0.9195585
[2,] -0.7285833  0.3929532
> sum(diag(C))
[1] 9
```

What matrix operations correspond to adding new observations or variables to datasets? In the following 'r' in 'rbind' stands for row, 'c' in 'cbind' stands for column.

```
> A
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8    5
[4,]    4    9    4
[5,]    5   10    8
> B
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
```

```
[3,]    3    6    9   12
> rbind(A,B[,1])
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8    5
[4,]    4    9    4
[5,]    5   10    8
[6,]    1    2    3
> cbind(B,t(A))
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    1    4    7   10    1    2    3    4    5
[2,]    2    5    8   11    6    7    8    9   10
[3,]    3    6    9   12   11   12    5    4    8
```

At this point we have learned how to define matrices (within R) and work with them, and (most importantly) we have met the help command '?' which is followed by a function name. A typical technique for learning how to work with R involves first verbalizing the intention (i.e., make a matrix) before conducting an online search (search 'R define matrix'). The search should quickly provide you with a function name ('matrix') and then you can use R to learn the details of the function ('?matrix').

Exercise 3.1. *Create two matrices, both with* 15 *rows and* 5 *columns. Combine them using 'rbind'. Combine them using 'cbind'. Make sure to label and define your creations using* $<-$ *or* $=$*. Display the results.*

Exercise 3.2. *An internet search can help with many aspects of R. Search for the symbol (not simply *) used for conducting matrix multiplication within R. Create simple two-by-two matrices and multiply them. What happens if you use only *?*

3.2 Accuracy of simulations

In this section we start conducting some basic simulations. We can simulate a random sample of size n from any distribution that is known to R. For example we can simulate samples of size $n = 10, 100, 1000, 10000$ from a standard normal distribution as follows.

```
> x_10=rnorm(10)
> x_100=rnorm(100)
> x_1000=rnorm(1000)
> x_10000=rnorm(10000)
```

Each x_n is a vector of n independent and randomly drawn points from a standard normal distribution. r stands for random, and $norm$ stands for the normal distribution, which defaults to

the standard normal distribution. In place of $norm$ we could have used exp for exponential, $unif$ for uniform, etc. Lets focus on x_{100} for a moment. Figure 3.1 shows a histogram for this simulated data. The command for creating such a histogram is

```
> hist(x_100)
```

We can modify the histogram so the area is equal to one. The modified histogram is displayed in Figure 3.2. In order to modify the histogram type

```
> hist(x_100,freq=FALSE)
```

A quick note regarding '=TRUE' and or '=FALSE': This jargon appears when an argument can take only two values. In this case 'freq=TRUE' commands for a typical histogram, where the height of a box is the number of observations (frequency). 'freq=FALSE' commands for the

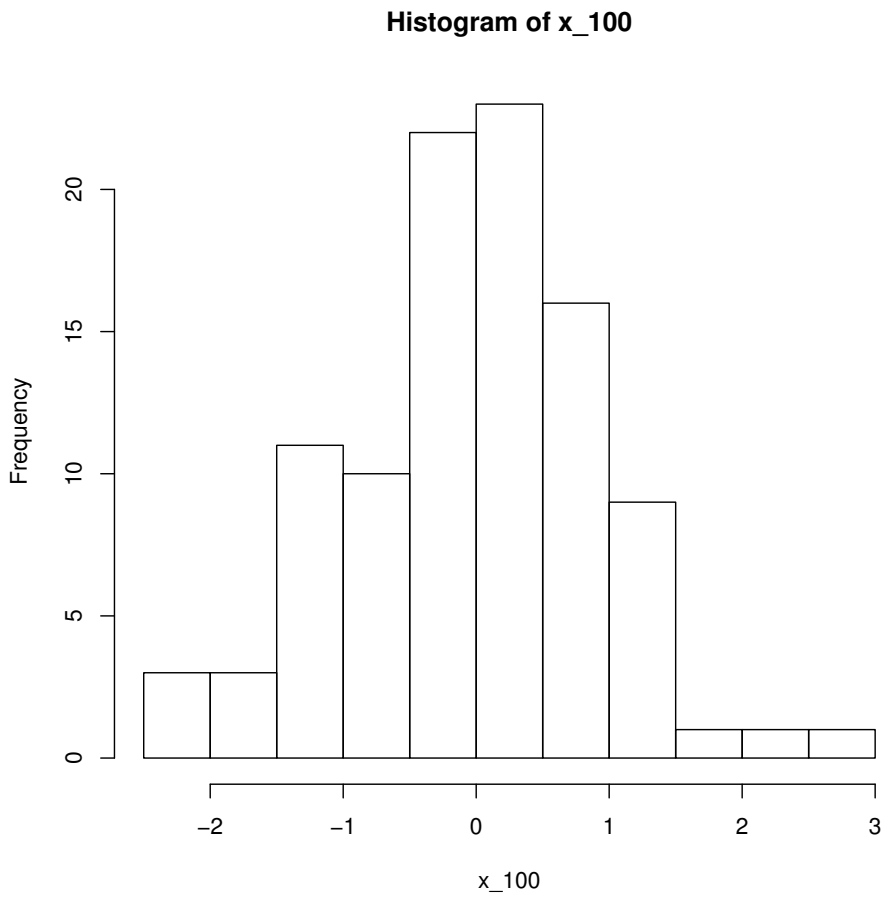


Figure 3.1. Histogram for a simulated random sample of 100 observations from a standard normal distribution

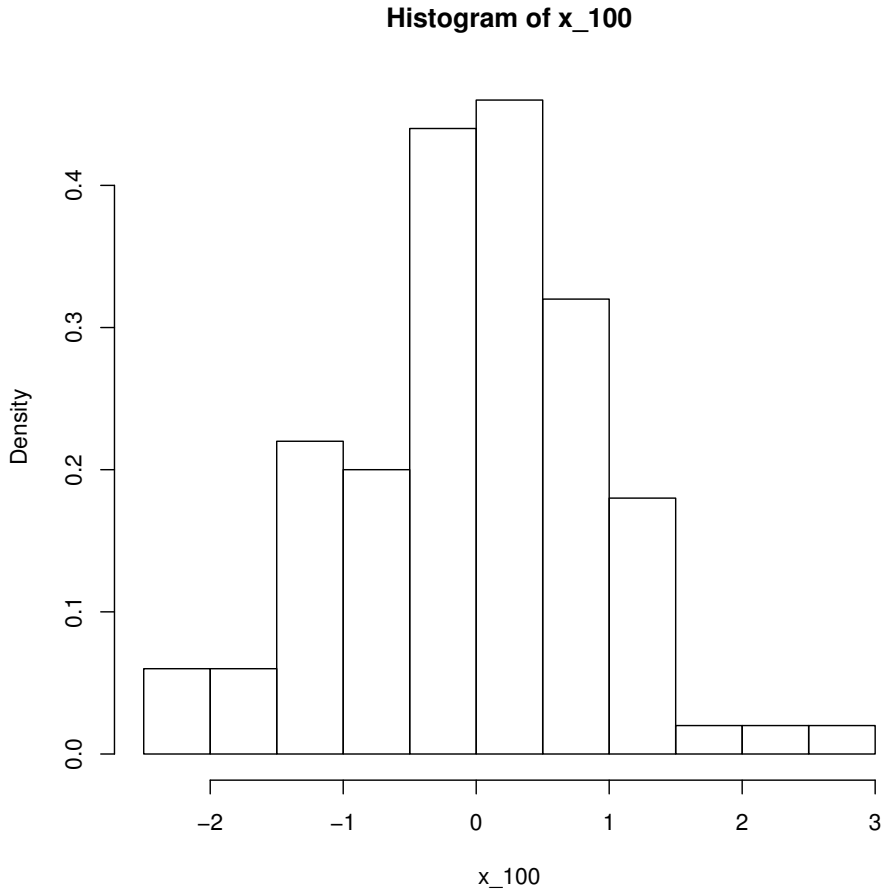


Figure 3.2. A modified histogram (area=1) for a simulated random sample of 100 observations from a standard normal distribution

alternative histogram, where the y-axis is scaled so that the histogram has total area equal to one. Additionally, we can use the sample to construct a kernal density estimate. For reading on the theory of kernal density estimation consult [15]. The default kernel for any sample is a standard normal kernel.

Here is the relevant command.

```
> plot(density(x_100))
```

The resulting output is displayed in Figure 3.3 To get a feel for how simulation accuracy depends on the simulated sample size n, and to illustrate some useful plotting features of R, we plot modified histograms and overlayed standard normal curves for each of the above $X_n : n = 10, 100, 1000, 10000$, all in the same graphic. To do this we prespecify that we would like four plots in the same graphic, in a 2 by 2 fashion.

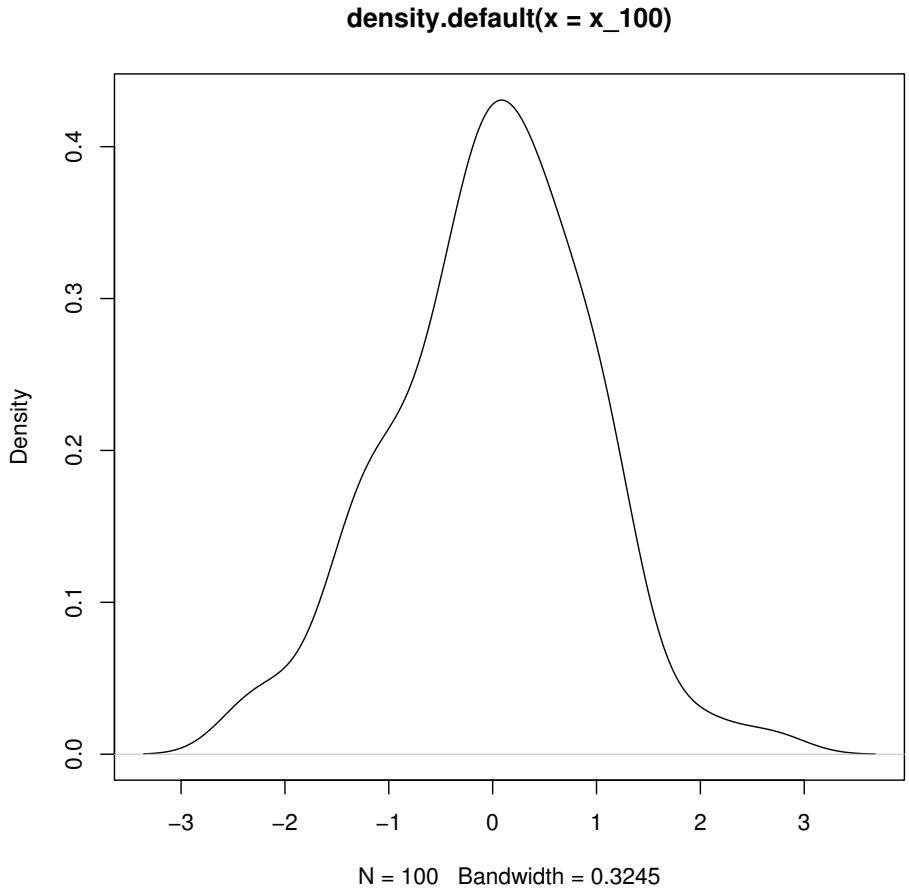


Figure 3.3. A simulated (n=100, kernel=normal) standard normal density

```
par(mfrow=c(2,2))
```

Then we proceed with the plots, adding $add = TRUE$ to create the overlay, and making sure to specify how much of the normal curve we would like to graph with the 'xlim' command.

```
> hist(x_10,freq=FALSE)
> plot(dnorm,add=TRUE,xlim=c(-3,3))
> hist(x_100,freq=FALSE)
> plot(dnorm,add=TRUE,xlim=c(-3,3))
> hist(x_1000,freq=FALSE)
> plot(dnorm,add=TRUE,xlim=c(-4,4))
> hist(x_10000,freq=FALSE)
> plot(dnorm,add=TRUE,xlim=c(-5,5))
```

The results are shown in Figure 3.4. For $n \geq 1000$ we seem to have fairly good accuracy, but as can be seen by repeating this procedure, and examining how the histograms fluctuate, it takes

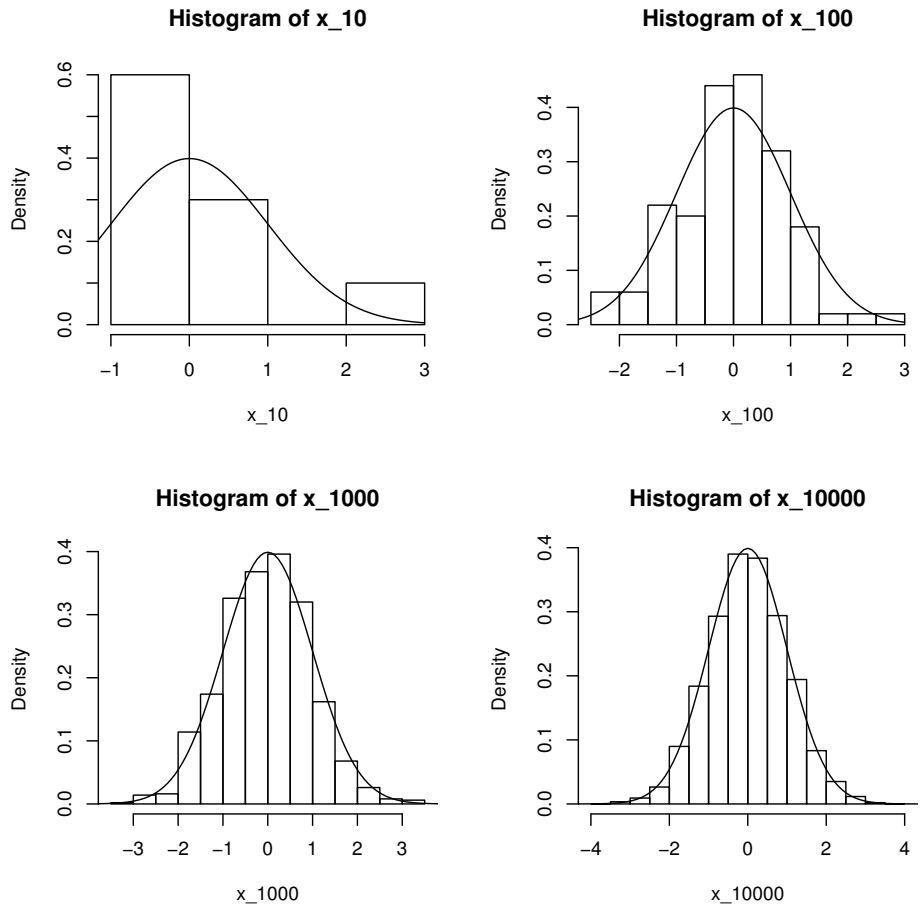


Figure 3.4. Normalized histograms for $x_n : n = 10, 100, 1000, 10000$ and overlayed standard normal curves

$n = 10000$ or more for the histograms to repeatedly look the same for each and every simulation. While the larger the n the better, how large is large enough will certainly depend upon many factors (population distribution, goal of the simulation, etc). Here we simply note that for $n = 1000$ it seems that we can get a pretty good picture of the distribution through simulation.

Exercise 3.3. *Use the internet to learn how to call ('norm' for normal distributions, 'exp' for exponential distributions, etc) three distributions of your choice within R. Repeatedly simulate random samples of various sizes from each distribution. Plot your results. How large must a simulated sample be in order to accurately represent the true theoretical distribution?*

3.3 Simulating central order statistics

The well-known central limit theorem states the following.

Theorem 3.1. (Central Limit Theorem) *Let $X_1, X_2, \ldots$ be an infinite sequence of independent, identically distributed random variables, each with a mean μ and a finite, positive variance σ^2. As $n \to \infty$*

$$S_n = \frac{\sum_{i=1}^{n}(X_i - \mu)}{\sqrt{n}\sigma} \to^d N(0,1)$$

A loose interpretation is that large-sample means have a bell-shaped density, centered around the population mean. In this section we illustrate that this holds, not only for the sample mean, but for the sample median, and other central quantiles as well. In what follows we use matrices and the extremely useful 'apply' command, as opposed to writing loops.

We begin with a sample of size ten from the exponential distribution.

```
> x_10=rexp(10)
> x_10
 [1] 0.96908339 1.54353661 0.17288383 0.39199554 2.63484761 0.28751601
 [7] 0.93252701 1.52728164 1.42060506 0.09562766
```

We then compute the sample mean.

```
> mean(x_10)
[1] 0.9975904
```

It is important to remind ourselves that this is just a single instance of the mean. If we were to do this procedure many times, we could plot all the results (each and every sample mean), and then make a histogram or a kernal density estimate to approximate the true distribution of this particular (average of 10 standard exponentials) sample mean. With a sample size of just ten will the sample mean be normally distributed? See problem 3.4.

We plan to simulate 1,000 different means, each coming from a sample of 1,000 standard exponential distributions. One way to do this is to use the 'apply' function. This function acts on a matrix (its first argument). The second argument is binary: 1 for rows and 2 for columns. The third argument is the name of a function, such as 'mean' that operates on vectors. The specified function will be applied to either the rows or columns of the matrix, returning a vector of results.

The following code computes the mean of each column, resulting in a simulated sample of 1000 means.

```
> x = rexp(1000000)
> M = matrix(x,nrow=1000)
> m = apply(M,2,mean)
```

'm' is then our simulated sample. Let us take a look at the kernel density estimate. The plot is obtained with

```
> plot(density(m))
```

and it is displayed in Figure 3.5. This illustrates how the sample mean is often distributed in a bell shaped fashion. What about the sample median? Let us simulate. We proceed as in the above case, but with median instead of mean. The commands are

```
x = rexp(1000000)
M = matrix(x,nrow=1000)
m = apply(M,2,median)
plot(density(m))
```

and the output is displayed in Figure 3.6. Thinking of the median as the center order statistic,

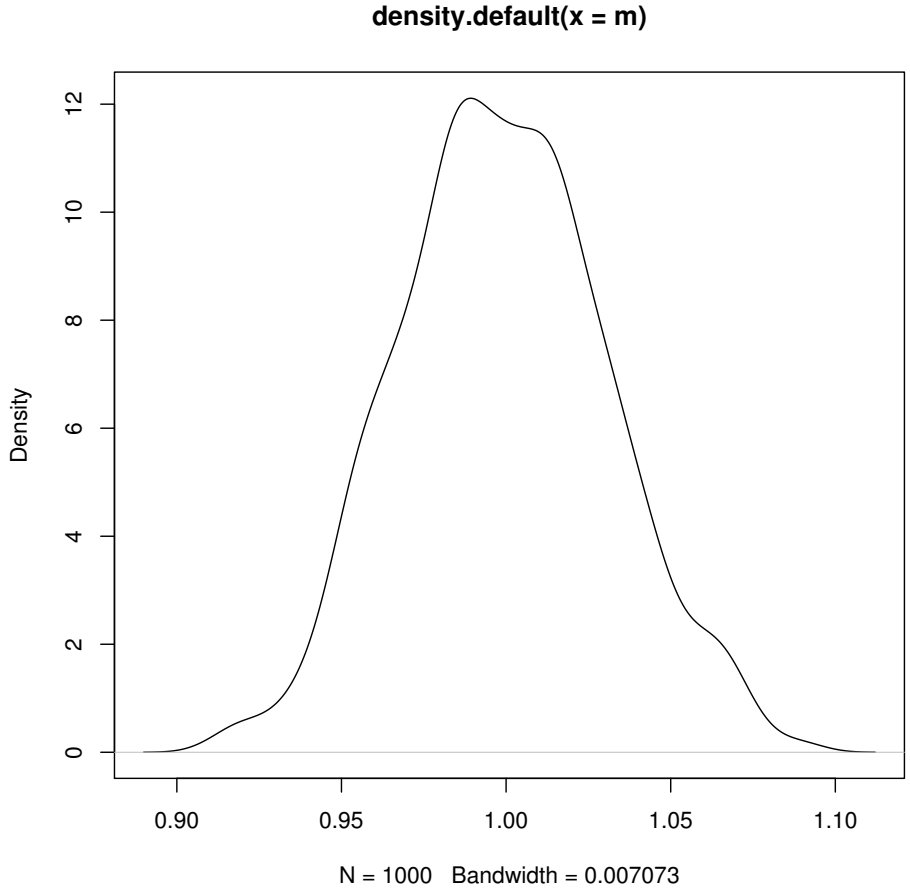


Figure 3.5. A kernel density estimate (standard normal kernal) for the density of $\bar{X}$, the mean of 1,000 standard exponential random variables

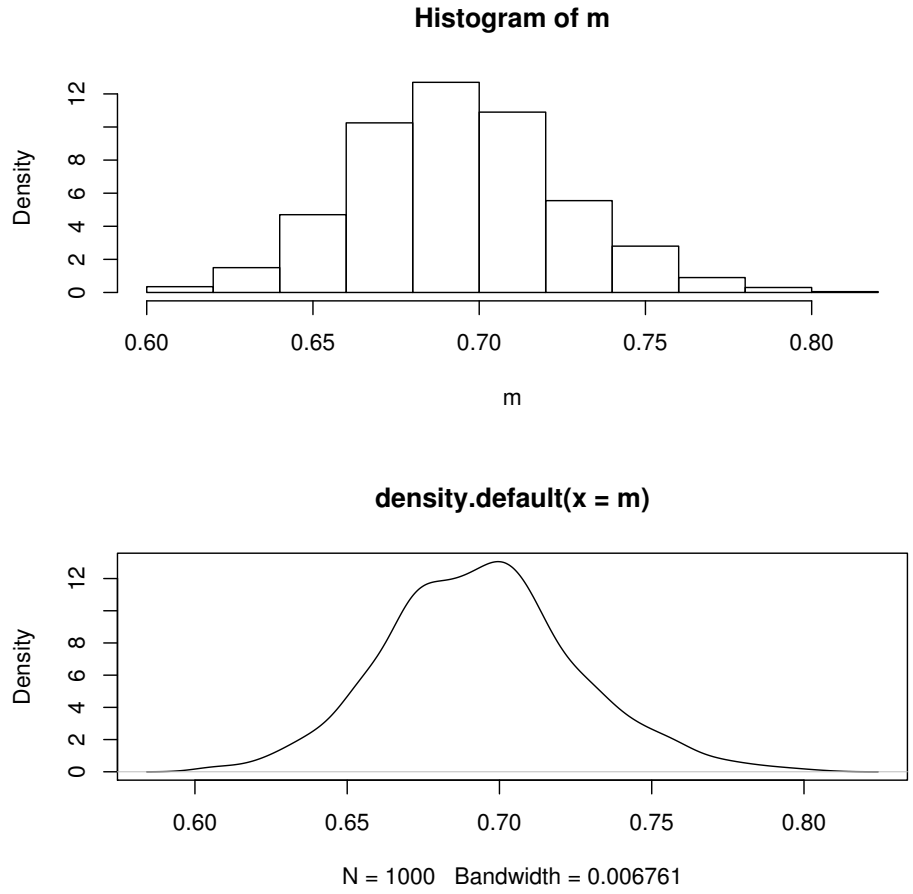


Figure 3.6. A histogram and kernel density estimate (standard normal kernal) for the density of the median of 1,000 standard exponential random variables

we have seen (at least in the case of 1000 standard exponentials) that this statistic has a bell shaped distribution. In fact, there are theorems (see Bain and Engelhardt, Theorem 7.51 [4]) showing that for a wide class of populations, the central order statistics, or the percentiles, are asymptotically normal. So we expect to simulate bell-shaped distributions. In what follows we simulate, not just the median, but 19 different percentiles (5%,10%,...,95%), all at once, each again coming from 1,000 simulated, standard exponential points.

This is a good place for learning how to define new functions within R. Our goal is to compute a matrix M as above, but not to apply 'mean' or 'median' to the columns. Rather, we would like to apply the 'quantile' function to the columns. The 'quantile' function simultaneously computes all of the specified (as additional arguments) percentiles of a given vector (primary argument). The percentiles can be thought of as the quantiles (or approximations thereof) from the sample

distribution F_n. However, R can't deal with 'apply(M,2,quantile)' because the additional arguments for 'quantile' have yet to be specified. A quick fix involves defining a new function.

```
> qbyfives <- function(x) quantile(x,c(.05,.1,.15,.2,
.25,.30,.35,.4,.45,.5,.55,.6,.65,.7,.75,.8,.85,.9,.95))
```

'qbyfives' is the name of the new function. 'function(x)' specifies that 'qbyfives' is a function and that this new function has a single argument, namely the dummy variable 'x' standing for the data to which 'quantile' will be applied. A space after 'function(x)' marks where the new function definition begins. The definition spans two lines just for clarity of presentation; it is

'quantile(x,c(.05,.1,.15,.2,.25,.30,.35,.4,.45,.5,.55,.6,.65,.7,.75,.8,.85,.9,.95)'

and can certainly be written on a single line within R.

With 'qbyfives' already defined we type 'apply(M,2,qbyfives)' to compute a vector of percentiles for each column. That is we have a two-dimensional result, 19 rows (one for each percentile) and all of the 1000 columns. We label this result with 'm'.

```
> m <- apply(M,2,qbyfives)
```

To visualize 'm' we would like to plot a kernal density estimate for each row. This requires more coding on our part. We need two more functions. The first acts on a vector and plots a density, the second acts on a matrix and will apply the first function to each row. We call the first function 'dplot' for density plot, and the second function 'Mplot' for matrix plot.

```
> dplot<-function(x) plot(density(x),main=NA,xlab=NA,ylab=NA)
> Mplot<-function(x) apply(x,1,dplot)
```

'Mplot' is then the final product and will plot the rows of a matrix. Remember, 'x' is just a dummy variable. In 'dplot' it stands for a vector. In 'Mplot' it stands for a matrix. It's whatever you have designed your function to act on.

Returning now to our goal of simulating many percentiles at once, the following code creates a kernal density estimate for each of the specified percentiles of a sample of 1000 standard exponentials, and by using 'par(mfrow=c(5,4)' plots them all simultaneously on the same graphic.

```
> x<-rexp(1000000)
> M<-matrix(x,nrow=1000)
> qbyfives <- function(x) quantile(x,c(.05,.1,.15,.2,
.25,.30,.35,.4,.45,.5,.55,.6,.65,.7,.75,.8,.85,.9,.95))
> m<-apply(M,2,qbyfives)
> dplot<-function(x) plot(density(x),main=NA,xlab=NA,ylab=NA)
> Mplot<-function(x) apply(x,1,dplot)
> par(mfrow=c(5,4))
> Mplot(m)
```

To view the resulting graphic see the supplementary material.

We have demonstrated the plausability that the central (even out to the extremes of the 5th and 95th) percentiles of an exponential distribution are asymptotically normal. What if we go all the way to the 1st and 99th percentiles?

```
> x<-rexp(1000000)
> M<-matrix(x,nrow=1000)
> qbyones <- function(x) quantile(x,c(.01,.02,.03,.04,
.05,.06,.07,.08,.09,.1,.9,.91,.92,.93,.94,.95,.96,.97,.98,.99))
> m<-apply(M,1,qbyones)
> dplot<-function(x) plot(density(x),main=NA,xlab=NA,ylab=NA)
> Mplot<-function(x) apply(x,1,dplot)
> par(mfrow=c(5,4))
> Mplot(m)
```

The resulting output can be seen in the supplementary material and shows that we still obtain asymptotic normality. Only the 1st percentile has begun to skew, perhaps due to the sample size of only 1000 and the boundary at zero. We expect boundaries to skew the results.

A uniform is bounded both below and above. The following code simulates the specified quantiles for a standard uniform.

```
> x=runif(1000000,0,1)
> M<-matrix(x,nrow=1000)
> qbyones <- function(x) quantile(x,c(.01,.02,.03,.04,
.05,.06,.07,.08,.09,.1,.9,.91,.92,.93,.94,.95,.96,.97,.98,.99))
> m<-apply(M,1,qbyones)
> dplot<-function(x) plot(density(x),main=NA,xlab=NA,ylab=NA)
> Mplot<-function(x) apply(x,1,dplot)
> par(mfrow=c(5,4))
> Mplot(m)
```

As before, the results are viewable in the supplementary material, and this time there is some skewing.

As a final experiment we substitute the fat-tailed Cauchy distribution for the uniform distribution. The commands are

```
> x<-rcauchy(1000000)
> M<-matrix(x,nrow=1000)
> qbyones <- function(x) quantile(x,c(.01,.02,.03,.04,
.05,.06,.07,.08,.09,.1,.9,.91,.92,.93,.94,.95,.96,.97,.98,.99))
> m<-apply(M,1,qbyones)
> dplot<-function(x) plot(density(x),main=NA,xlab=NA,ylab=NA)
> Mplot<-function(x) apply(x,1,dplot)
> par(mfrow=c(5,4))
> Mplot(m)
```

and the results, viewable in the supplementary material, show plenty of skewed graphs.

It seems that the convergence for the asymptotic normality of the percentiles, at least when the population distribution is Cauchy, can be quite slow, especially when the percentile is extreme.

Exercise 3.4. *Conduct simulations to determine how large sample sizes must be so that random samples from exponential distributions have means that are distributed approximately normally.*

Exercise 3.5. *Choose a distribution and conduct simulations to test the normality of the central order statistics. The sample sizes should be how large? (hard) Can you write R code that will efficiently plot many independent, simulated kernel density estimates for the central order statistics?*

3.4 Simulating extreme order statistics

Extreme order statistics are not the same as the 99th percentile, the 1st percentile, or any percentile. The difference becomes clear when thinking asymptotically. When $n = 100$ then $X_{n:n}$ will behave in a manner similar to the 99th percentile. As n grows $X_{n:n}$ remains the maximum of the sample, and thus we can expect for it to grow itself. On the other hand we expect the 99th percentile to remain in the vicinity of the 99th quantile for the population. The comparison between $X_{1:n}$ and the $1st$ percentile is similar. Asymptotically, the same can be said for $X_{n-1:n}, X_{n-2,n}$, etc and $X_{2,n}, X_{3,n}$, etc. They are extreme order statistics and are asymptotically distinct from any percentile, even a 99.999 or 0.001 percentile. For simplicity, we will focus on $X_{n:n}$ as a prototypical extreme order statistic. Our goal is to determine its asymptotic distribution.

As long as X is not constant,

$$P[X_1, X_2, ..., X_n < \sup\{X\}] > 0,$$

which underscores how $X_{n:n}$ is itself a random variable with its own distribution. Upon consideration, the fact that $X_{n:n}$ has its own distribution is not surprising. However, asymptotically, when it does not degenerate it must be one of three types (Bain and Engelhardt, Chapter 7, [4]). This intriguing result is sometimes referred to as the Fisher-Tippet-Gnedenko Theorem. In what follows we will conduct simulations to illustrate the three types.

3.4.1 Type 1

When sampling from exponential, normal, or even log-normal distributions, the maximal order statistic will have an asymptotic distribution of type 1 (Bain and Engelhardt, Chapter 7.8 [4]). We illustrate with some simulations.

```
> e=rexp(1000000)
> M_e=matrix(e,nrow=1000)
> m_e=apply(M_e,2,max)
>
> n=rnorm(1000000)
> M_n=matrix(n,nrow=1000)
> m_n=apply(M_n,2,max)
>
> ln=rlnorm(1000000)
> M_ln=matrix(ln,nrow=1000)
> m_ln=apply(M_ln,2,max)
```

'm_e','m_n', and 'm_{ln}' denote random samples of maximums for exponential, normal, and log-normal, respectively. In each case the maximum is the maximum of samples of size $1,000$, and again for each case we have $1,000$ instances of the max. This simulation should provide a decent picture of each limiting distribution. We plot the estimated densities, making use of an optional argument, namely 'main', which can be used to add titles to plots.

```
> par(mfrow=c(3,1))
> plot(density(m_e),main="Exponential")
> plot(density(m_n),main="Normal")
> plot(density(m_ln),main="LogNormal")
```

The resulting plots can be viewed in the supplementary material. After standardization, and in the limit, all three would converge to a type 1 distribution, also known as a Gumbell distribution.

Note that these graphs are not standardized, after standardization, and in the limit, all three would converge to a type 1 distribution, also known as a Gumbell distribution.

3.4.2 Type 2

When sampling from a uniform or beta distribution, the maximal order statistic will have an asymptotic distribution of type 2 (Bain and Engelhardt, Chapter 7.8 [4]). Again, we illustrate with some simulations.

The setup and procedure is analogous to that which we used in the Type 1 section.

```
> u=runif(1000000)
> M_u=matrix(u,nrow=1000)
> m_u=apply(M_u,2,max)
>
> bl=rbeta(n=1000000,2,2)
> M_bl=matrix(bl,nrow=1000)
> m_bl=apply(M_bl,2,max)
>
> bh=rbeta(n=1000000,.5,.5)
> M_bh=matrix(bh,nrow=1000)
> m_bh=apply(M_bh,2,max)
```

```
>
>
> par(mfrow=c(3,1))
> plot(density(m_u),main="Uniform")
> plot(density(m_bl),main="Beta(2,2)")
> plot(density(m_bh),main="Beta(.5,.5)")
```

The resulting plots, viewable in the supplementary material show yet-to-be standardized approximations to type 2 distributions, also known as Frechet distributions.

3.4.3 Type 3

When sampling from a Cauchy or slash distribution, the maximal order statistic will have an asymptotic distribution of type 3 (Bain and Engelhardt, Chapter 7.8 [4]). This is the fat-tailed case. The slash distribution is defined in the following way.

Definition 3.2. *(Slash Distribution) Let Z denote a standard normal random variable and U an independent standard uniform random variable. The slash random variable is defined as*

$$S = Z/U$$

Use of the slash distribution within R requires the loading of a package, in this case the 'VGAM' package. Once installed, we can use the following code to produce the following simulated graphs.

```
> c=rcauchy(1000000)
> M_c=matrix(c,nrow=1000)
> m_c=apply(M_c,2,max)
>
> require(VGAM)
> s=rslash(1000000)
> M_s=matrix(s,nrow=1000)
> m_s=apply(M_s,2,max)
>
>
> par(mfrow=c(2,1))
> plot(density(m_c),main="Cauchy")
> plot(density(m_s),main="Slash").
```

The graphs are displayed as part of the supplementary material and they illustrate type 3, or Reversed Weibell distributions.

Exercise 3.6. *If X is distributed as a Gamma ($X \sim GAM(\theta,\kappa)$) then the asymptotic distribution of $X_{n,n}$ will be a type 1 distribution. Illustrate this fact by conducting a simulation. When plotting your results, include a title by using 'main'.*

Exercise 3.7. *Examine the x-axis from figures* **??** *and* **??**. *The latter has a much larger range. Explain.*

Exercise 3.8. *Simulate $X_{n:n}$ for various n when X has a normal distribution. Simulate $X_{n:n}$ for various n when X has a Cauchy distribution.*

CHAPTER 4

MAXIMUM LIKELIHOOD ESTIMATION WITH MAPLE

Maple is a computer algebra system that was developed at the University of Waterloo in Ontario, Canada. In this section we use Maple to numerically solve optimization problems. We begin with an example that demonstrates the need for numerical, optimization techniques within statistics. We then learn how the techniques can be implemented within Maple in order to solve real-world, statistical problems.

4.1 Numerics

Consider the following example that demonstrates the need for numerics.

Imagine that k different universities are involved in an effort to precisely estimate a physical constant μ. Teams of physicists at each university use their own experimental methods and complete their own experiment n_i times. Each university thus has n_i measurements of the physical constant μ. Even if we assume independence and normality it is not clear how we should combine all the measurements into a single estimate for μ. The difficulty arises because we have not assumed equal variances across the different experiments at the different universities. We need to estimate $k+1$ parameters, namely the k group variances and the common mean μ. Note that the mean is common across groups because it is a physical constant, presumed to be constant throughout the Universe. We will solve this problem using numeric techniques.

In order to demonstrate the need for numerics clearly, without getting bogged down in notation, we temporarily assume that $k = 2$. We can then express the sample from the first university as $\{x_1, ..., x_{n_1}\}$ and the sample from the second university as $\{y_1, ..., y_{n_2}\}$. As mentioned above, the nature of the problem assumes that $\mu_1 = \mu_2 = \mu$, however σ_1^2 need not equal σ_2^2. The likelihood function is thus a function of three unknown parameters:

$$L(\mu, \sigma_1^2, \sigma_2^2) = (2\pi)^{-(n_1+n_2)/2} \sigma_1^{-n_1} \sigma_2^{-n_2} \exp\left(-\frac{1}{2} \sum_{i=1}^{n_1} [(x_i - \mu)/\sigma_1]^2 - \frac{1}{2} \sum_{j=1}^{n_2} [(y_j - \mu)/\sigma_2]^2 \right).$$

For the purpose of maximizing we can drop the $(2\pi)^{-(n_1+n_2)/2}$ term. Note that for a fixed μ we then have

$$L(\sigma_1^2,\sigma_2^2)=\sigma_1^{-n_1}\exp\left(-\frac{1}{2}\sum_{i=1}^{n_1}[(x_i-\mu)/\sigma_1]^2\right)\sigma_2^{-n_2}\exp\left(-\frac{1}{2}\sum_{j=1}^{n_2}[(y_j-\mu)/\sigma_2]^2\right), \tag{4.1}$$

which will be maximized at

$$\widehat{(\sigma_1^2,\sigma_2^2)}=(\hat{\sigma}_1^2,\hat{\sigma}_2^2),$$

where

$$\hat{\sigma}_1^2=\sum_{i=1}^{n_1}(x_i-\mu)^2/n_1 \text{ and } \hat{\sigma}_2^2=\sum_{j=1}^{n_2}(y_j-\mu)^2/n_2 \tag{4.2}$$

are the familiar maximum likelihood estimators for variance from the single sample setting. Thus, we have lowered the dimension of the problem. The maximul likelihood estimator $\widehat{(\mu,\sigma_1^2,\sigma_2^2)}$ must satisfy

$$\widehat{(\mu,\sigma_1^2,\sigma_2^2)}=(\hat{\mu},\hat{\sigma}_1^2,\hat{\sigma}_2^2),$$

where $\hat{\sigma}_i^2$ are as defined in 4.2 and it remains to solve for $\hat{\mu}$, the maximizer of

$$L(\mu)=\hat{\sigma}_1^{-n_1}\exp\left(-\frac{1}{2}\sum_{i=1}^{n_1}[(x_i-\mu)/\hat{\sigma}_1]^2\right)\hat{\sigma}_2^{-n_2}\exp\left(-\frac{1}{2}\sum_{j=1}^{n_2}[(y_j-\mu)/\hat{\sigma}_2]^2\right). \tag{4.3}$$

Upon substitution of the definitions from 4.2 into 4.3 we get some simplification. $\hat{\mu}$ must be that which maximizes

$$L(\mu)=\left(\sum_{i=1}^{n_1}(x_i-\mu)^2/n_1\right)^{-n_1/2}e^{-n_1^2/2}\left(\sum_{j=1}^{n_2}(y_j-\mu)^2/n_2\right)^{-n_2/2}e^{-n_2^2/2}.$$

The exponential parts can be dropped and the $1/2$ factors in the remaining exponents do not influence optimization, and thus they can be dropped as well. After generalizing to allow for more than two experiments, the result is the following theorem.

Theorem 4.1. *Assume that k different experiments have been carried out to measure the same quantity μ, and that for $i=1,2,...,k$ each experiment has been run n_i times, resulting in the data $\{x_{1,1},...,x_{1,n_1},.....,x_{k,1},...,x_{k,n_k}\}$. If for all $i=1,2,...,k$ we have $X_i=^d N(\mu,\sigma_i^2)$ and if all observations are independent, then the maximum likelihood estimators are given by*

$$\hat{\mu}=\arg\max_{\mu}\prod_{i=1}^{k}\left(\sum_{j=1}^{n_i}(x_{i,j}-\mu)^2/n_i\right)^{-n_i} \tag{4.4}$$

and

$$\hat{\sigma}_i^2=\sum_{j=1}^{n_i}(x_{i,j}-\hat{\mu})^2/n_i.$$

If the sample sizes are equal then for the purposes of optimization we can neglect the exponents in (4.4) and the resultant expression to maximize is a degree $2k$ polynomial. Critical points can be determined by differentiation and solving for the roots of the resulting, degree $2k-1$ polynomial. However, we might suspect from results in modern algebra that an analytic solution for the roots of such a polynomial, especially when k is large, might be too much to hope for. Keep in mind also that unusual samples can effect the form of such a polynomial. Thus, even in the simpler case of equal sample sizes the analytic approach bogs down. A practical approach is to use numerical techniques and that is what we do throughout this chapter.

4.2 Basic optimization

In this section we present two methods for finding where a polynomial attains its maximum. These methods will be illustrated in Maple.

First we need to define the function. The following illustrates the proper syntax (note the semicolon). The polynomial f is defined.

```
>f := x -> -x^4+20*x^3+1*x^2-1*x+2;
```

It can be evaluated at a point x_0 by typing the following.

```
>f(x_0);
```

Before starting a search for a maximum it is a good idea to plot the function.

```
>plot(f(x));
```

The resulting plot is displayed in Figure 4.1. The plot can be misleading however. Make sure your graphic captures all salient features that are relevant to optimization. This can be accomplished by adjusting the range of x-values, as in the following example.

```
>plot(f(x),x=-10..25);
```

The resulting plot is displayed in Figure 4.2. The graphic shows us that the maximum occurs near $x = 15$. To find a more exact value we first differentiate $f(x) = -x^4 + 20 * x^3 + 1 * x^2 - 1 * x + 2$ and label it g,

```
>g:=x -> diff(f(x),x);
```

The result is $g(x) = -4 * x^3 + 60 * x^2 + 2 * x - 1$ which we set equal to zero and then solve using 'solve' and 'fsolve'. The output is shown in Figure 4.3. 'solve' produces three algebraic expressions for each of the three roots. 'fsolve' produces numeric approximations for each of the three roots.

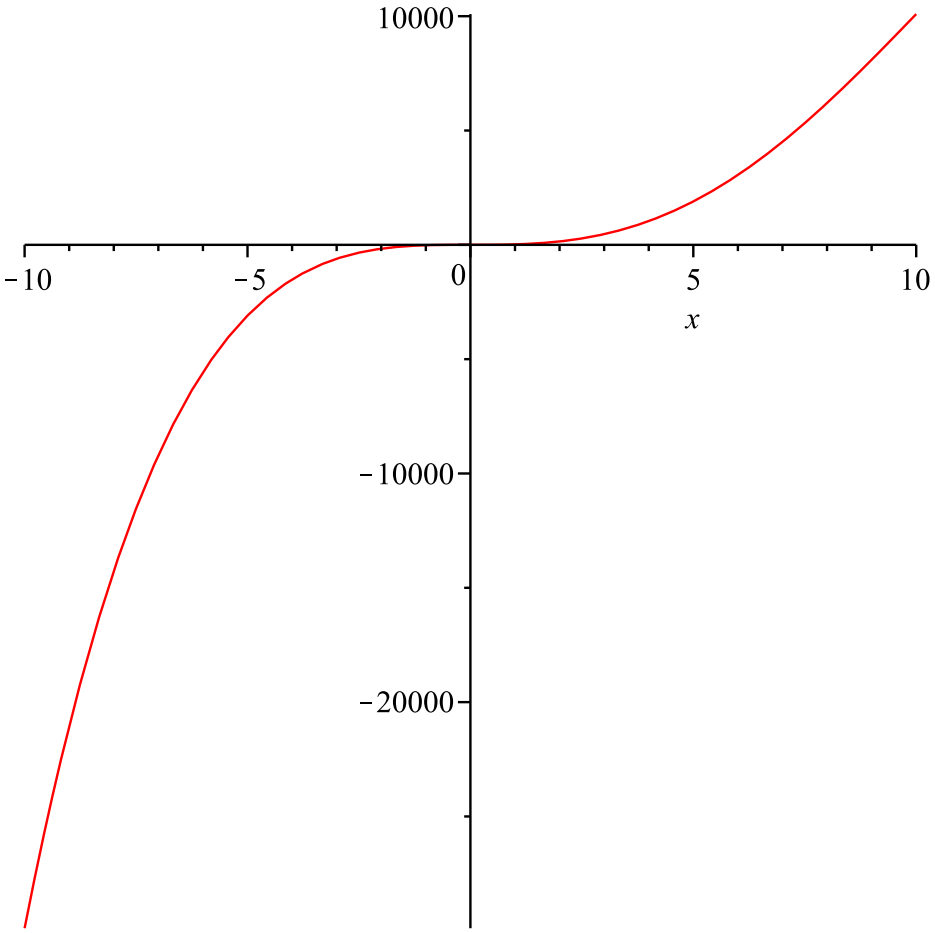


Figure 4.1. Maple's default plot for $f(x) = -x^4 + 20 * x^3 + 1 * x^2 - 1 * x + 2$

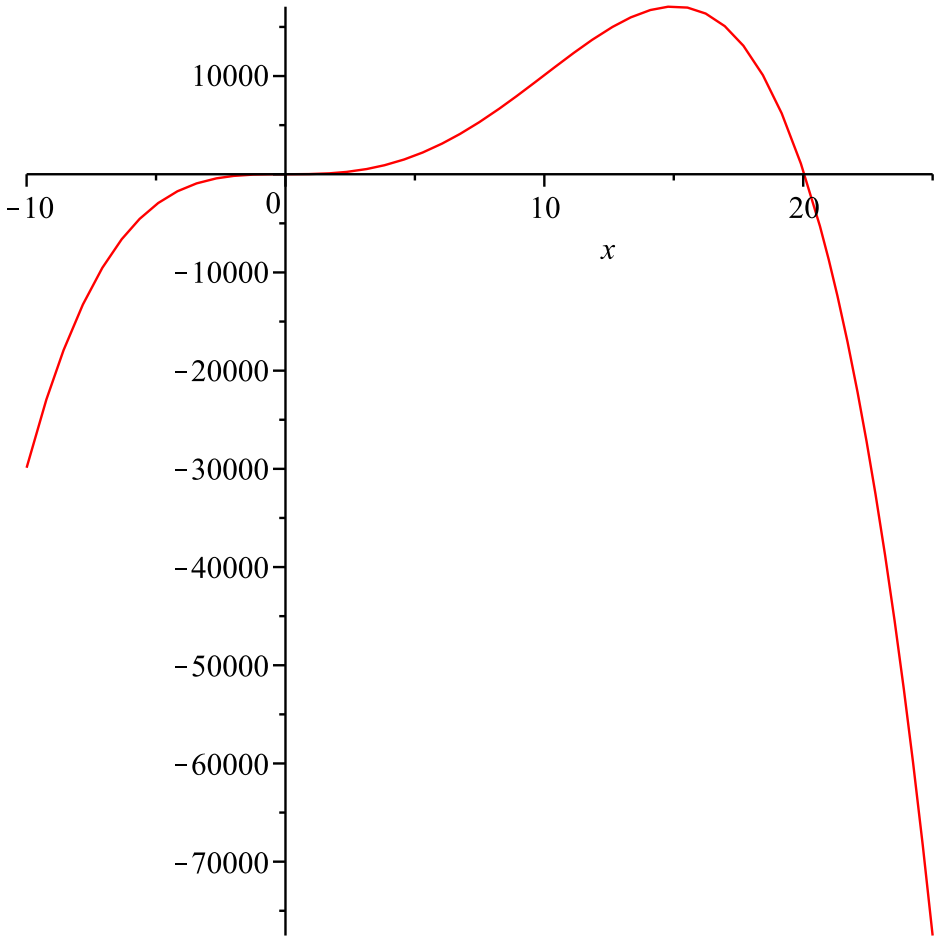


Figure 4.2. Maple's default plot for $f(x) = -x^4 + 20 * x^3 + 1 * x^2 - 1 * x + 2$ after specifying a range of x values

```
> solve(g(x)=0,x);
(1/6)*(27243+(3*I)*166263^(1/2))^(1/3)+151/
(27243+(3*I)*166263^(1/2))^(1/3)+5, -
(1/12)*(27243+(3*I)*166263^(1/2))^(1/3)-(151/2)/
(27243+(3*I)*166263^(1/2))^(1/3)+5+
((1/2)*I)*3^(1/2)*((1/6)*(27243+(3*I)*166263^(1/2))^(1/3)-151/
(27243+(3*I)*166263^(1/2))^(1/3)), -
(1/12)*(27243+(3*I)*166263^(1/2))^(1/3)-(151/2)/
(27243+(3*I)*166263^(1/2))^(1/3)+5-
((1/2)*I)*3^(1/2)*((1/6)*(27243+(3*I)*166263^(1/2))^(1/3)-151/
(27243+(3*I)*166263^(1/2))^(1/3))
> fsolve(g(x)=0,x);
-.1460374859, .1138818193, 15.03215567
```

Figure 4.3. Maple commands and output related to 'solve'

We are more interested in the results of 'fsolve'. We have seen that $\arg\max_x f(x) \approx 15$. According to our calculations this can be refined to

$$\arg\max_x f(x) \approx 15.03215567.$$

Next, we search for the same x that maximizes $f(x) = -x^4 + 20 * x^3 + 1 * x^2 - 1 * x + 2$, not by differentiation, but by using Maple's optimization package. We simply type

```
>with(Optimization);
>Maximize(f(x));
```

Figure 4.4 displays the output.

Exercise 4.1. *At http://www.maplesoft.com learn how to work with logarithms and exponentials, and then use Maple to define, plot and maximize* $h(x) = \log(x) - e^x$ *on* $(0,\infty)$. *Where does the maximum occur?*

4.3 Vectors and matrices

In this section we learn how to define vectors and matrices within Maple. We also learn how to define functions on the entries of vectors and matrices.

A vector v is defined with the 'linalg' package. For example,

```
> with(linalg);
> v:=vector([7,9,3]);
```

produces the vector $(7,9,3)$. We can then call an entry by specifying its position.

```
> v[2]
```

returns the value 9. Matrices are similar. For example,

```
> M:=Matrix(2,3,[1,2,3,4,5,6]);
```

produces the matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$

To isolate an entry of the matrix we specify the row and the column. For example,

```
> with(Optimization);
[ImportMPS, Interactive, LPSolve, LSSolve, Maximize, Minimize,
NLPSolve, QPSolve]
> Maximize(f(x));
[17087.4669232051310, [x = 15.0321556665687677]]
```

Figure 4.4. Using Maple's optimization package and the command 'Maximize'

```
> M[2,1];
```

will produce the number 4.

We will have the need to compute indexed sums and products. We can use 'add' and 'mul' (for multiply) respectively. For example, to sum the entries of our previously defined vector v we type

```
> add(v[i],i=1..3);
```

Likewise, to multiply the entries of our vector v we type

```
> mul(v[i],i=1..3);
```

These operations can be combined as follows.

```
> mul(add(M[i,j],i=1..2),j=1..3);
```

The above command adds the entries within each column and then multiplies the results.

Exercise 4.2. *This exercise uses select data (for simplicity) from the famous Michelson-Morley experiment. The data has been obtained from the 'morley' dataset within R. Three different experiments have been run* 10 *times each, resulting in the following observed values for the speed of light in millions of feet per second:*

$$\begin{pmatrix} 850 & 960 & 880 \\ 740 & 940 & 880 \\ 900 & 960 & 880 \\ 1070 & 940 & 860 \\ 930 & 880 & 720 \\ 850 & 800 & 720 \\ 950 & 850 & 620 \\ 980 & 880 & 860 \\ 980 & 900 & 970 \\ 880 & 840 & 950 \end{pmatrix}.$$

Use Maple to combine all the data into a single point estimate for the speed of light. You may assume independence and normality but not equal variances across experiments. Hint: see Theorem 4.1.

4.4 Optimization over higher-dimensional sets

As preparation for this section we first complete the exercise from the previous section in detail using Theorem 4.1. With an equal number of runs, $n = 10$, for each of the three experiments, $k = 3$, the conclusion of Theorem 4.1 can be stated as

$$\hat{\mu} = \arg\max_{\mu} \prod_{j=1}^{3} \left(\sum_{i=1}^{10} (x_{i,j} - \mu)^2 / 10 \right)^{-10}, \tag{4.5}$$

where $x_{i,j}$ is the entry in the ith row and the jth column of

$$M = \begin{pmatrix} 850 & 960 & 880 \\ 740 & 940 & 880 \\ 900 & 960 & 880 \\ 1070 & 940 & 860 \\ 930 & 880 & 720 \\ 850 & 800 & 720 \\ 950 & 850 & 620 \\ 980 & 880 & 860 \\ 980 & 900 & 970 \\ 880 & 840 & 950 \end{pmatrix}.$$

This matrix M can be defined in Maple with

```
> M:=Matrix(10,3,[850,740,900,1070,930,850,950
,980,980,880,960,940,960,940,880,800,850,880,
900,840,880,880,880,860,720,720,620,860,970,950]);
```

The expression from 4.5 can then be defined in Maple as follows:

```
> L := mu -> mul(((add((M[i,j]-mu)^(2),i=1..10)/10)^(-10)),j=1..3);
```

Figure 4.5 plots the function. The plot does not display the region of the graph of

$$L(\mu) = \prod_{j=1}^{3} \left(\sum_{i=1}^{10} (x_{i,j} - \mu)^2 / 10 \right)^{-10}$$

where the maximum occurs. Based on the numerical values for the data we next plot L for $\mu : \mu \in (700, 1000)$. The resulting graphic is displayed in Figure 4.6. We conclude from the plot that $\hat{\mu} \approx 875$. In order to obtain a more precise numeric value for $\hat{\mu}$ we use Maple's optimization package and the 'Maximize' command. Figure 4.7 shows how we are initially met with minor errors. This is typical when working on optimization problems. Ingenuity is often required to find a particular fix for the specific problem at hand. Figure 4.8 shows our attempt to get around the problem by explicitly specifying an initial starting point. There is still a small problem however which is remedied by multiplying the function by the factor 10^{118}. The result is a function

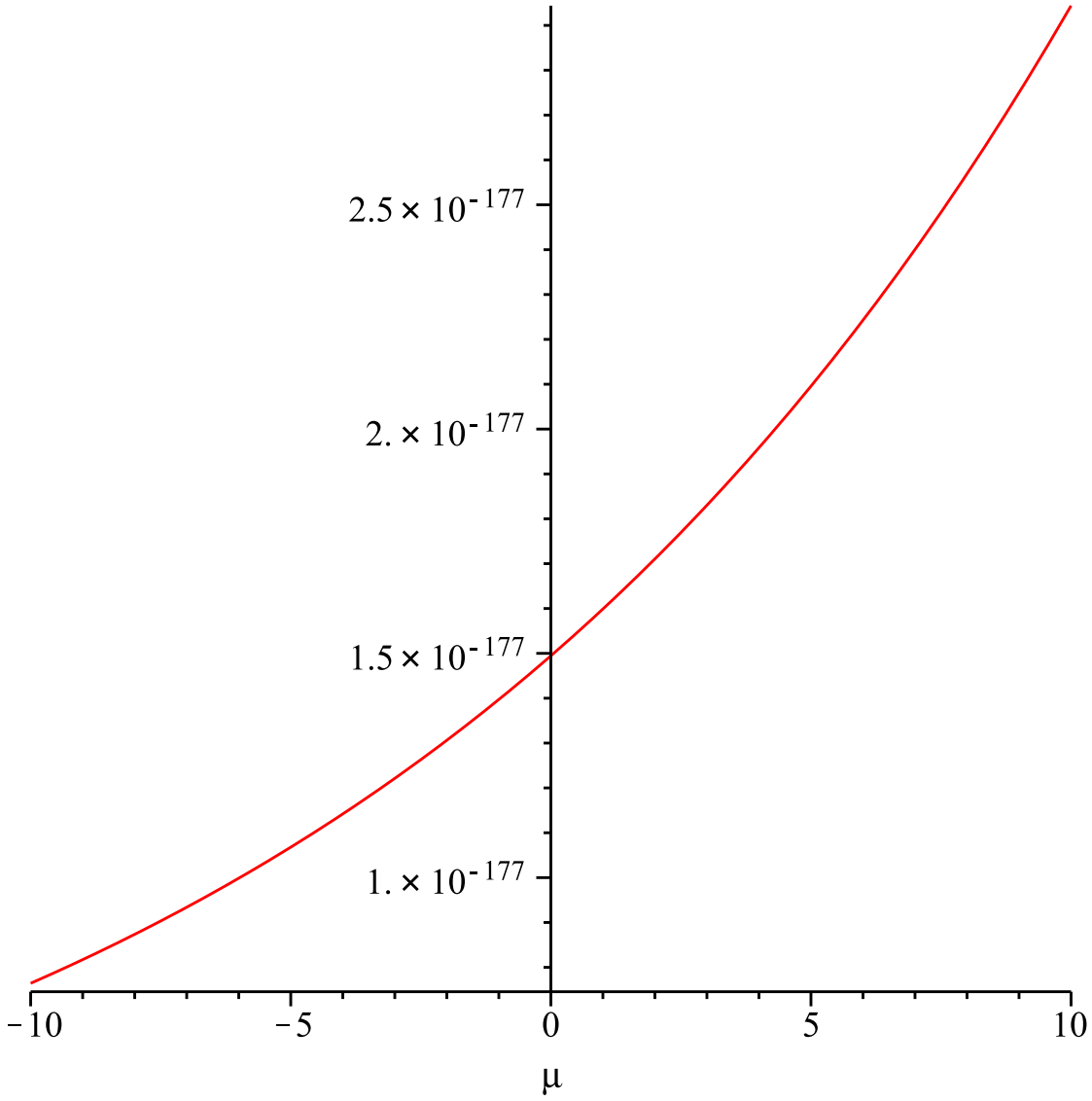


Figure 4.5. Maple's default plot for $L(\mu) = \prod_{j=1}^{3}\left(\sum_{i=1}^{10}(x_{i,j}-\mu)^2/10\right)^{-10}$, obtained via 'plot(L(mu));'

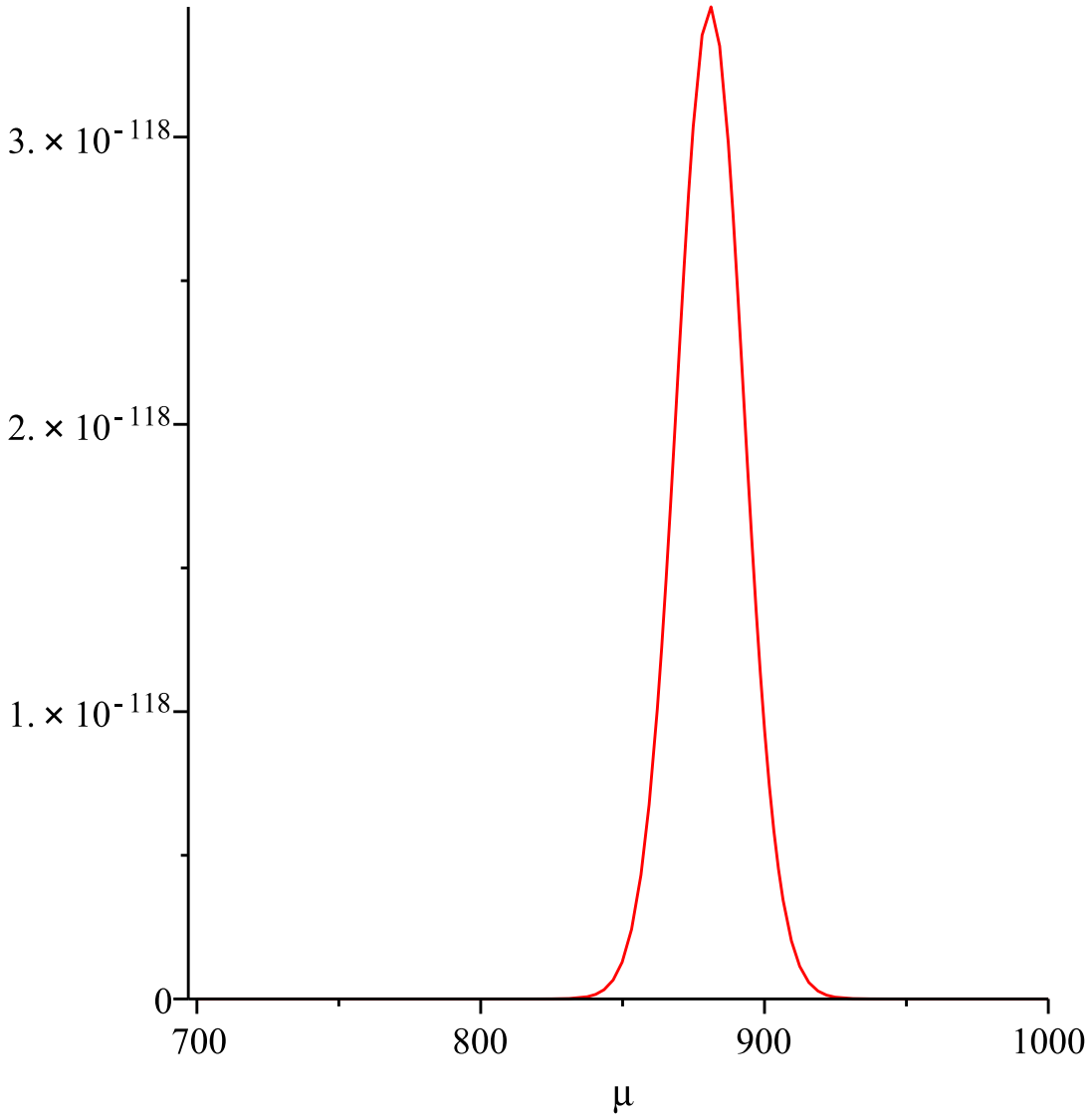


Figure 4.6. This plot of $L(\mu) = \prod_{j=1}^{3} \left(\frac{\sum_{i=1}^{10} (x_{i,j} - \mu)^2}{10} \right)^{-10}$, obtained via 'plot(L(mu),mu=700..1000);', shows that $\hat{\mu} \approx 875$.

```
> with(Optimization);
[ImportMPS, Interactive, LPSolve, LSSolve, Maximize, Minimize,
NLPSolve, QPSolve]

> Maximize(L(mu));
Error, (in Optimization:-NLPSolve) the objective gradients at the
initial point are too small
```

Figure 4.7. 'Maximize' follows the gradient to the maximum, but the extremely small gradient at the initial point results in an error.

```
> Maximize(L(mu),initialpoint={mu=870});
Error, (in Optimization:-NLPSolve) the objective gradients at the
initial point are too small
```

Figure 4.8. Even with a well-chosen initial point the gradient is too small, due to the fact that the function values are miniscule, on the order of 10^{-118}.

that can be maximized with Maple's maximize command and the output is displayed in Figure 4.9. Supposing that this computed value $\hat{\mu} \approx 880.848$ is acceptable, we can now use this value to estimate the variances. Maple's output is displayed in Figure 4.10. At this point we have used Theorem 4.1 to obtain maximum likelihood estimates for the four parameters:

$$(\hat{\mu} = 880.848114270578208, \hat{\sigma_1^2} = 8098.845699, \hat{\sigma_2^2} = 7829.023069, \hat{\sigma_3^2} = 8790.896670).$$

Keep these estimates in mind as references, as we now present an alternative method that will estimate all four parameters at once.

Assuming independence and normality and again with $x_{i,j}$ denoting the measurement associated with the ith run of the jth experiment, the likelihood function can be written as

$$L(\mu, \sigma_1^2, \sigma_2^2, \sigma_3^2) = (2\pi)^{-(n_1+n_2+n_3)/2} \sigma_1^{-n_1} \sigma_2^{-n_2} \sigma_3^{-n_3} \exp\left(-\frac{1}{2} \sum_{j=1}^{3} \left(\sum_{i=1}^{n_j} [(x_{i,j} - \mu)/\sigma_j]^2 \right) \right).$$

Note that the variances have been used as the parameters even though their square roots appear in the expression of the function. The relevant part of the function can be defined in Maple as shown in Figure 4.11. A few preliminary attempts to maximize the function fail as shown in Figure 4.12. The error states that the objective gradients at the initial point are too small. Perhaps if we wisely choose our initial point closer to the maximum, the gradients will be large enough for 'Maximize' to work. In what follows we use the grand mean as our initial guess for μ and the

```
> Maximize(10^(118)*L(mu),initialpoint={mu=870});

[3.45430970338556919, [mu = 880.848114270578208]]
```

Figure 4.9. Scaling the function by a factor of 10^{118} allows Maple to work with the computed gradients and find numerical values for the maximum value and the maximizing point.

```
> s2s=(add((M[i,1]-880.848114270578208)^(2),i=1..10)/10,add((M[i,2]-
880.848114270578208)^(2),i=1..10)/10,add((M[i,3]-
880.848114270578208)^(2),i=1..10)/10);

s2s = (8098.845699, 7829.023069, 8790.896670)
```

Figure 4.10. Using $\hat{\mu} = 880.848114270578208$ to compute the maximum likelihood estimates for the variances, following the second half of Theorem 4.1.

```
> L := (mu,v1,v2,v3) -> (sqrt(v1)*sqrt(v2)*sqrt(v3))^(-10)*exp(-
(1/2)*(add(((M[i,1]-mu)/sqrt(v1))^2,i=1..10)+add(((M[j,2]-
mu)/sqrt(v2))^2,j=1..10)+add(((M[k,3]-mu)/sqrt(v3))^2,k=1..10)));

L := proc (mu, v1, v2, v3) options operator, arrow; exp(-
(1/2)*add((M[i, 1]-mu)^2/sqrt(v1)^2, i = 1 .. 10)-(1/2)*add((M[j, 2]-
mu)^2/sqrt(v2)^2, j = 1 .. 10)-(1/2)*add((M[k, 3]-mu)^2/sqrt(v3)^2, k
= 1 .. 10))/(sqrt(v1)^10*sqrt(v2)^10*sqrt(v3)^10) end proc
```

Figure 4.11. Defining the relevant part of the likelihood function using Maple

```
> with(Optimization);

[ImportMPS, Interactive, LPSolve, LSSolve, Maximize, Minimize,
NLPSolve, QPSolve]

> Maximize(L(mu,v1,v2,v3));

Error, (in Optimization:-NLPSolve) the objective gradients at the
initial point are too small

> Maximize(10^(118)*L(mu,v1,v2,v3));

Error, (in Optimization:-NLPSolve) the objective gradients at the
initial point are too small
```

Figure 4.12. Even with the extra factor of $10^{(118)}$ the gradients are too small

sample variances as our initial guesses for the variances. These quantities can be specified in Maple as shown in Figure 4.13: With our initial point wisely chosen we can now again attempt to Maximize. The output is shown in Figure 4.14. Unfortunately, Maple is still responding with an error message. In order to troubleshoot it's worthwhile to review our code. We defined

$$L(\mu,\sigma_1^2,\sigma_2^2,\sigma_3^2) =$$
$$(2\pi)^{-(n_1+n_2+n_3)/2}\sigma_1^{-n_1}\sigma_2^{-n_2}\sigma_3^{-n_3}e^{-\frac{1}{2}\left(\sum_{i=1}^{n_1}[(x_{i,1}-\mu)/\sigma_1]^2+\sum_{i=1}^{n_2}[(x_{i,2}-\mu)/\sigma_2]^2+\sum_{i=1}^{n_3}[(x_{i,3}-\mu)/\sigma_3]^2\right)}.$$

as a function of the variances. These variances are large (in the thousands) and could be causing problems for Maple. It is worth our effort to redifine the function in terms of the standard deviations (the square roots of the variances). This can be done in Maple as shown in Figure 4.15. Now with the numbers at a more manageable size, Maple can complete the maximization problem, as illustrated in Figure 4.16. Here is the resulting vector of estimates (rounded to four decimal digits):

$$(\hat{\mu} = 880.8478, \hat{\sigma_1} = 89.9928, \hat{\sigma_2} = 88.4814, \hat{\sigma_3} = 93.7601).$$

Compare it to the previous results obtained via Theorem 4.1:

$$(\hat{\mu} = 880.848114270578208, \hat{\sigma_1^2} = 8098.845699, \hat{\sigma_2^2} = 7829.023069, \hat{\sigma_3^2} = 8790.896670).$$

After squaring the estimated standard deviations, so that they are readily compared with the estimated variances, we see that both methods give comparable results.

Exercise 4.3. *An object falling from rest falls a vertical distance d as a function of time t. More specifically,*

$$d = \frac{1}{2}gt^2$$

where g is the acceleration due to gravity at or near the surface of the earth. This model was the basis for an experiment that was run three times by undergraduate students of physics at the University of Utah in 2003. Their resulting measurements of g were

$$\textit{for experiment one}\ (9.815, 9.714, 9.514).$$

A second experiment was carried out, this time rolling solid steel balls down an incline with angle θ. The transverse acceleration along the incline is related to g via

$$a = \frac{5}{7}g\sin\theta$$

```
> muguess := add(add(M[i,j],i=1..10),j=1..3)/30;

muguess := 2642/3

> v1guess := add(((M[i,1]-add(M[j,1],j=1..10)/10)^2/9),i=1..10);

v1guess := 26870/3

> v2guess := add(((M[i,2]-add(M[j,2],j=1..10)/10)^2/9),i=1..10);

v2guess := 78290/9

> v3guess := add(((M[i,3]-add(M[j,3],j=1..10)/10)^2/9),i=1..10);

v3guess := 87440/9
```

Figure 4.13. Defining educated guesses for the initial point

```
>
Maximize(L(mu,v1,v2,v3),initialpoint={mu=muguess,v1=v1guess,v2=v2gues
s,v3=v3guess});

Error, (in Optimization:-NLPSolve) the objective gradients at the
initial point are too small

>
Maximize(10^(118)*L(mu,v1,v2,v3),initialpoint={mu=875,v1=8098,v2=7829
.023069,v3=8790.896670});

Error, (in Optimization:-NLPSolve) no improved point could be found
```

Figure 4.14. Even with our wisely-choosen initial point the gradient is too small

```
> Ls := (mu,s1,s2,s3) -> (s1*s2*s3)^(-10)*exp(-(1/2)*(add(((M[i,1]-
mu)/s1)^2,i=1..10)+add(((M[j,2]-mu)/s2)^2,j=1..10)+add(((M[k,3]-
mu)/s3)^2,k=1..10)));

Ls := proc (mu, s1, s2, s3) options operator, arrow; exp(-
(1/2)*add((M[i, 1]-mu)^2/s1^2, i = 1 .. 10)-(1/2)*add((M[j, 2]-
mu)^2/s2^2, j = 1 .. 10)-(1/2)*add((M[k, 3]-mu)^2/s3^2, k = 1 ..
10))/(s1^10*s2^10*s3^10) end proc
```

Figure 4.15. Redefining the likelihood function as a function of standard deviations

```
>
Maximize(10^(118)*Ls(mu,s1,s2,s3),initialpoint={mu=muguess,s1=sqrt(v1
guess),s2=sqrt(v2guess),s3=sqrt(v3guess)});

Warning, limiting number of major iterations has been reached

[0.568543118407565908e53, [mu = 880.847875783519157, s1 =
89.9928687280392126, s2 = 88.4814006723392623, s3 =
93.7601714815641572]]
```

Figure 4.16. The maximum likelihood vector of parameters is obtained with Maple's maximize command

so that the distance (along the incline) depends on t according to

$$d = \frac{1}{2}\frac{7}{5}\frac{1}{\sin\theta}t^2$$

Using this model and running the experiment three times the students recorded the following additional measurements of g:

for experiment two $(5.7582, 5.7918, 5.6602)$.

If we assume that the measurement errors are normal and that all the runs are independent, but don't assume equal variances across experiments, then the theory of this chapter is applicable. However, we probably should not use it to estimate g in this case. Why not?

CHAPTER 5

CONFIDENCE INTERVALS AND HYPOTHESIS TESTING WITH SAS

SAS stands for Statistical Analysis System and refers to both the company SAS Institute Inc. in North Carolina and its software.

When you open a SAS session you will see multiple windows. An example is the 'Explorer' window as displayed in Figure 5.1. The remaining windows are pictured as part of the supplementary material.

The 'Log' window keeps track of what has been done in SAS. After submitting a program one can check the 'Log' window to see if the program ran correctly, or not.

Programs are written into the 'Program Editor' window.

There is also a 'Toolbox'.

Behind these windows we find two additional windows: The 'Results' window and the 'Output' window.

These six windows can be overwhelming at first but with some use one quickly becomes familiar with the setup.

5.1 Importing data into SAS

We start by importing a dataset into SAS. Consider the dataset saved as HockeySticks.csv. The 'csv' stands for comma separated values. We can load the dataset into SAS as follows.

Multiple windows contain the header *File*, click on it and then *Import Data*. This will open the Import Wizard where there are self-explanatory instructions for pointing and clicking toward the goal of importing the data into SAS. First we select '.csv' as shown in Figure 5.2. Next we locate the already-saved dataset on our computer as shown in Figure 5.3. Then we save the dataset within SAS as shown in Figure 5.4, and also instruct SAS to create a file of the sequence of commands used to import the data as shown in Figure 5.5. Note that we should create our own folder to store such files. Here we have done this and titled the folder as 'SAS'. Instruct SAS to save the file in your newly created folder. The file that has been created is six lines of code, ready to be used in SAS:

Figure 5.1. SAS: Explorer window

```
PROC IMPORT OUT= WORK.HOCKEYSTICKS
            DATAFILE= "/home/1018/ma/knaeble/Desktop/HockeySticks.csv"
            DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
RUN;
```

This code can be typed into the Program Editor and then submitted (under 'run') and it will import the data. We have already done this however, using the mouse. We can check the SAS: Log to see if SAS successfully imported the data. In order to view the data click on *View* and then *Explorer*. Double click on *Libraries* and then *Work* (the library where we saved the data), and then double click on *HockeySticks* which will produce a table.

Exercise 5.1. *Access the dataset HockeySticks.csv from the course webpage and import it into SAS*

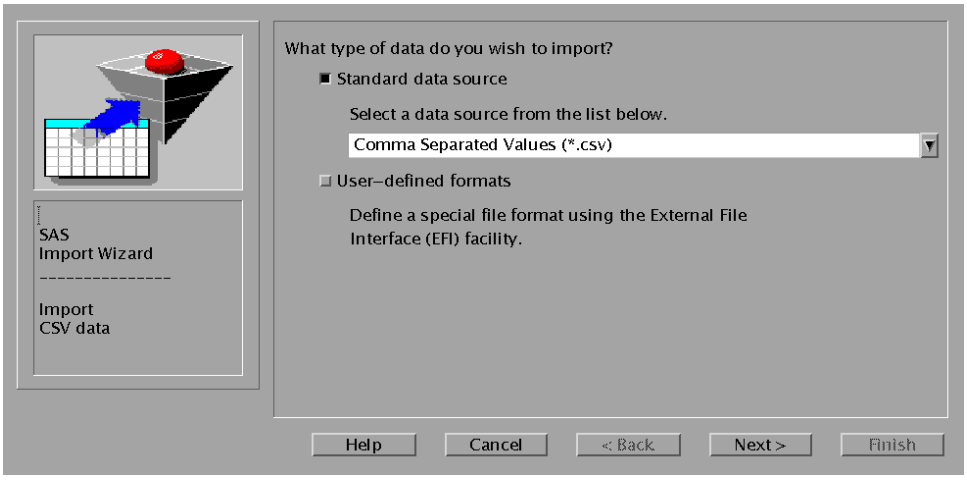


Figure 5.2. Selecting .csv

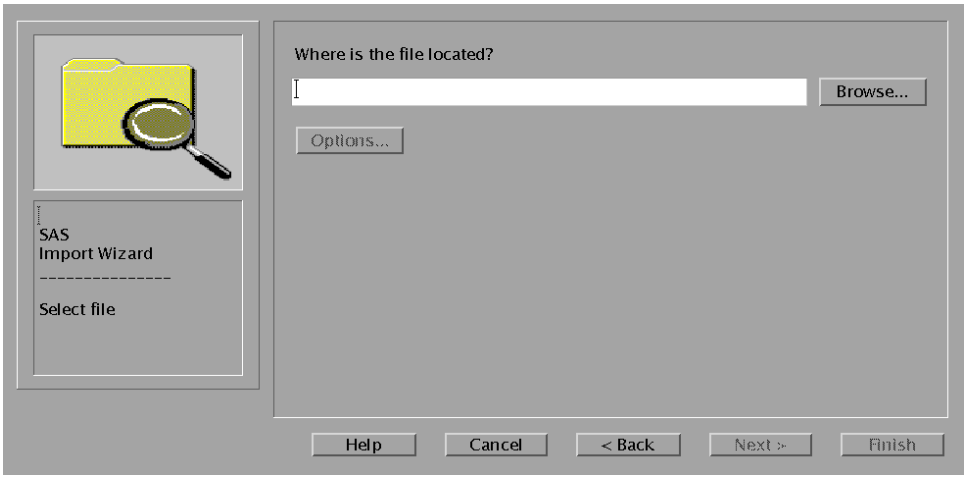


Figure 5.3. Specify where the data have been saved, use of 'Browse...' is OK

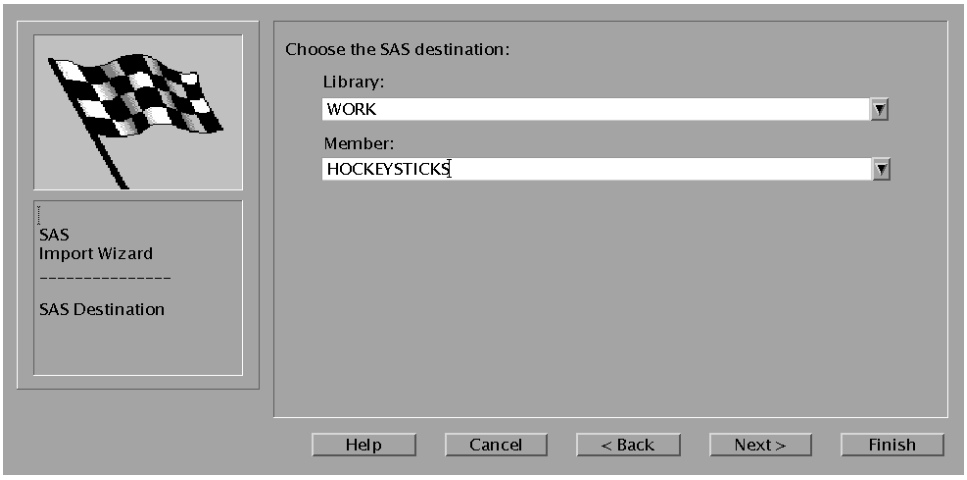


Figure 5.4. Within SAS we save the data in the library WORK and label it HOCKEYSTICKS

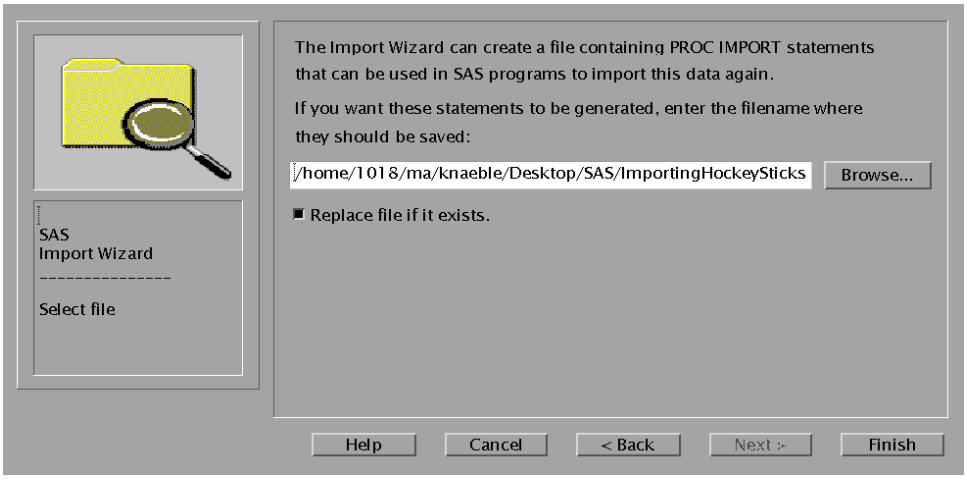


Figure 5.5. Saving the sequence of commands to a specified filename

5.2 Basic procedures in SAS

In the previous section we learned how to import data into SAS.

Specifically, we have a dataset named HockeySticks saved in the *Work* library. Its location can be seen through the *SAS: Explorer* window. Click on *Libraries* and then *Work*. In order to go back to the previous level click *View* and then *Up One Level*.

In the SAS: Program Editor we can call this data by typing Work.HockeySticks. In this section we will learn to perform basic procedures on the Hockey Stick data.

Our first endevour is to create a summary of descriptive statistics. This is done by typing the following lines of code into the Program Editor.

```
PROC MEANS DATA=Work.HockeySticks;
RUN;
```

This is only a two–line program. The program editor provides a place to edit many lines of code before they are submitted.

For example, suppose we had typed

```
PROC MEAN DATA=Work.HockeySticks;
RUN;
```

into the program editor. We submit this program by clicking on *Run* and then *Submit*. The result is an error as seen in the SAS: Log and shown here in Figure 5.6. SAS informs us that it could not carry out the program because the procedure MEAN was not found. This alerts us to our error,

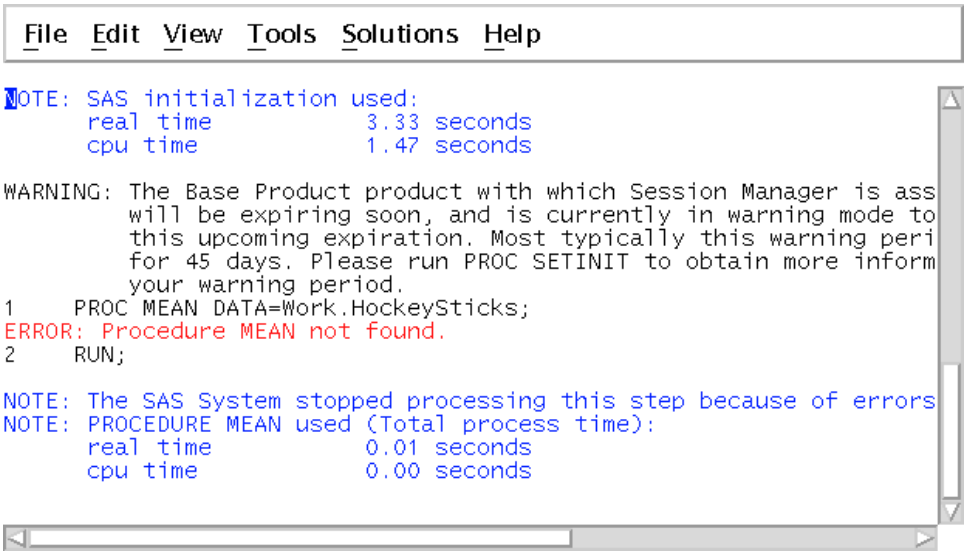


Figure 5.6. After submitting a program we can check the SAS: Log. Errors are shown in red.

the correct syntax was to specify MEANS not MEAN. Upon correcting this and submitting the program again we can check the log as displayed in Figure 5.7.

The program has been successfully run. The output window displays the results as shown in Figure 5.8. You might have to resize the window or center it. SAS nicely summarizes the data displaying the number of observations, the mean, the standard deviation, the minimum, and the maximum.

In the case of categorical data another useful command is PROC FREQ which works analogously to PROC MEANS but returns frequency statistics. Next we endevour to use SAS to create a histogram. PROC UNIVARIATE is the command to remember. The following shows all three lines of code.

```
PROC UNIVARIATE DATA=Work.HockeySticks;
   HISTOGRAM;
RUN;
```

After submitting this code the log should show a lack of errors and SAS will automatically display the histogram as shown in the supplementary material.

Exercise 5.2. *PROC MEANS automatically returns the number of observations, the mean, the standard deviation, the minimum, and the maximum of any variables of interest. By using PROC MEANS other statistics can be computed as well. To do this type the name of the statistic after you specify the data. Use this technique to compute the median of the HockeySticks data.*

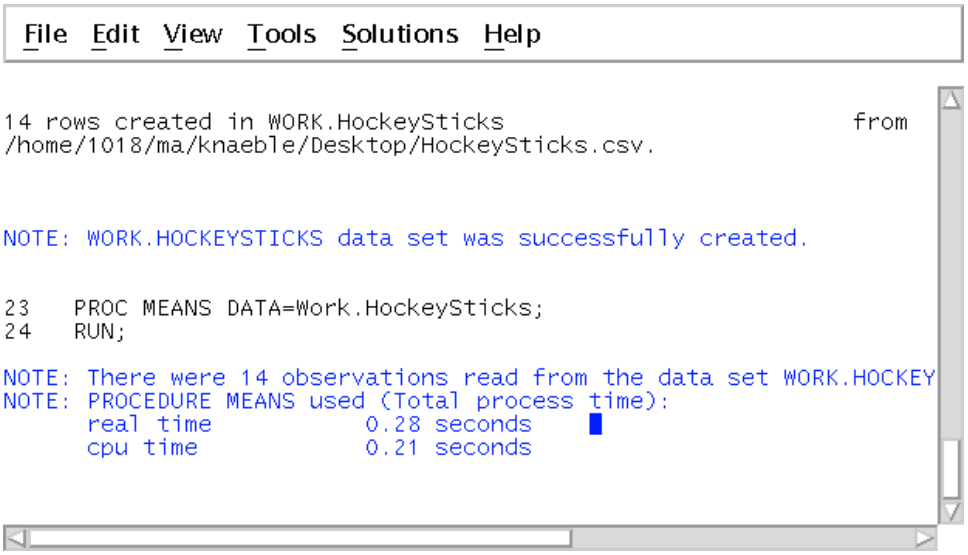


Figure 5.7. The program, copied in the log in black writing, has been successfully run. Blue writing summarizes what was done

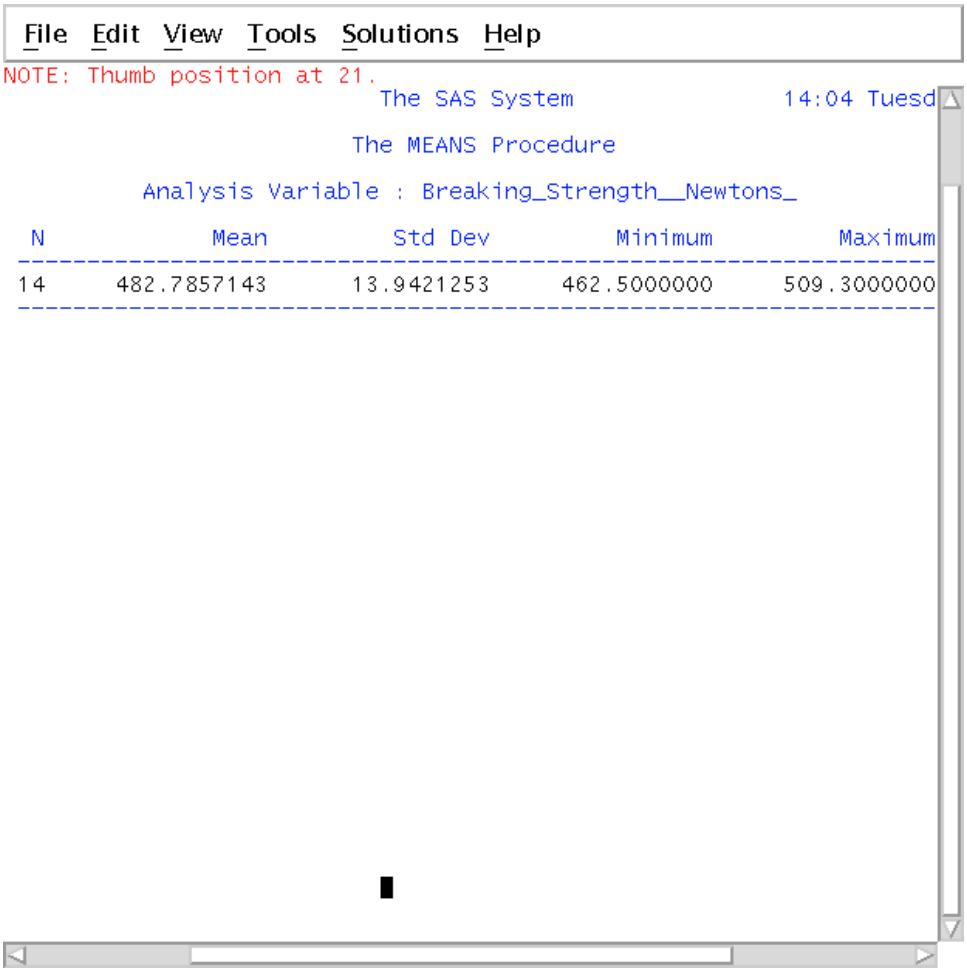


Figure 5.8. Output of the Proc Means procedure for our Hockey Sticks data.

5.3 Two sided T-test

In the next chapter we will deal with hypothesis testing in more detail. Here we simply state the following fact:

Theorem 5.1. *Let $X_1, ..., X_n$ be a random sample of independent observations with $X_i =^d N(\mu, \sigma^2)$. With*

$$S = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2)}{n-1}}$$

we then have

$$\frac{\bar{X} - \mu}{S/\sqrt{n}} =^d T(n-1).$$

Proof. See Theorem 8.4.3 of Bain and Engelhardt ([4]). □

$\frac{\bar{X}-\mu}{S/\sqrt{n}}$ is a pivotal quantity: it is distributed as a $T(n-1)$, which does not involve μ. The theorem allows us to determine $t_{1-\alpha/2}$ so that

$$\begin{aligned} 1-\alpha &= P\left(-t_{1-\alpha/2} < \frac{\bar{X} - \mu}{S_X/\sqrt{n}} < t_{1-\alpha/2}\right) \\ &= P\left(\bar{X} - t_{1-\alpha/2}S/\sqrt{n} < \mu < \bar{X} + t_{1-\alpha/2}S/\sqrt{n}\right), \end{aligned}$$

which specifies

$$(\bar{X} - t_{1-\alpha/2}S/\sqrt{n}, \bar{X} + t_{1-\alpha/2}S/\sqrt{n}) \tag{5.1}$$

as a $1-\alpha$ confidence interval for μ.

We now switch to the related notion of defining a hypothesis test of $H_0 : \mu = \mu_0$, where μ_0 is some fixed, real number. Under this null hypothesis Theorem 5.1 states that

$$\frac{\bar{X} - \mu_0}{S/\sqrt{n}} =^d T(n-1). \tag{5.2}$$

We can then compute

$$\frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

and reject the null if the resulting value is too extreme (as judged via the equality and T distribution from (5.2). In other words, our rejection region is symmetric and consists of the tails of $T(n-1)$. More precisely, we reject H_0 if

$$\left|\frac{\bar{x} - \mu_0}{s/\sqrt{n}}\right| > t_{1-\alpha/2}(n-1). \tag{5.3}$$

We can use SAS to simultaneously conduct such a hypothesis test and compute the associated confidence interval for μ.

```
PROC TTEST DATA=Work.HockeySticks;
RUN;
```

The output is shown in Figure 5.9. We can change the settings. For example, to test $H_0 : \mu_0 = 500$ we add $H0 = 500$ to the commands. To alter the confidence interval we specify α: $ALPHA = .1$ will result in a 90% confidence interval, $ALPHA = .01$ will result in a 99% confidence interval, etc. Here is an example of modified code.

```
PROC TTEST DATA=Work.HockeySticks HO=500 ALPHA=.1;
RUN;
```

The resulting output is displayed in Figure 5.10.

Exercise 5.3. *Why do you think that SAS includes a confidence interval for the mean as part of its t test output? Prove that $p < \alpha$ if and only if the confidence interval fails to contain μ_0.*

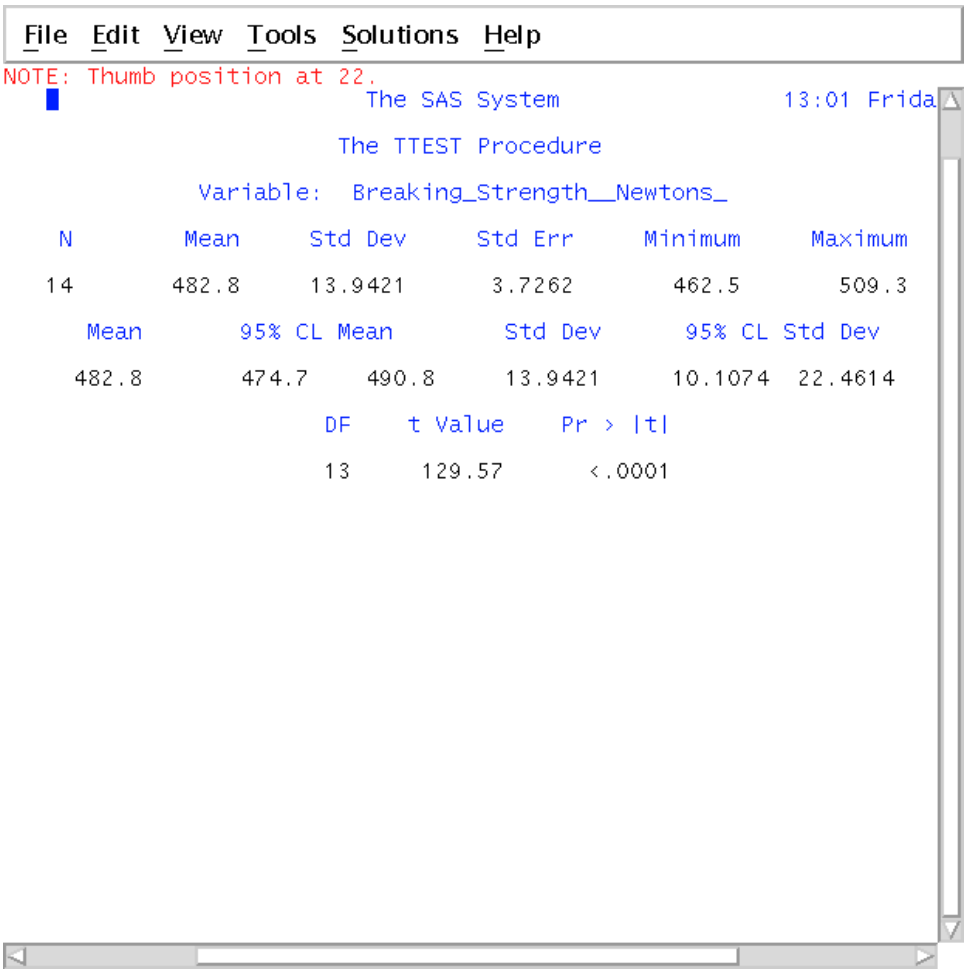


Figure 5.9. Results of our basic t test. Note that the default settings are 95% for the confidence interval and $\mu_0 = 0$ for the null hypothesis.

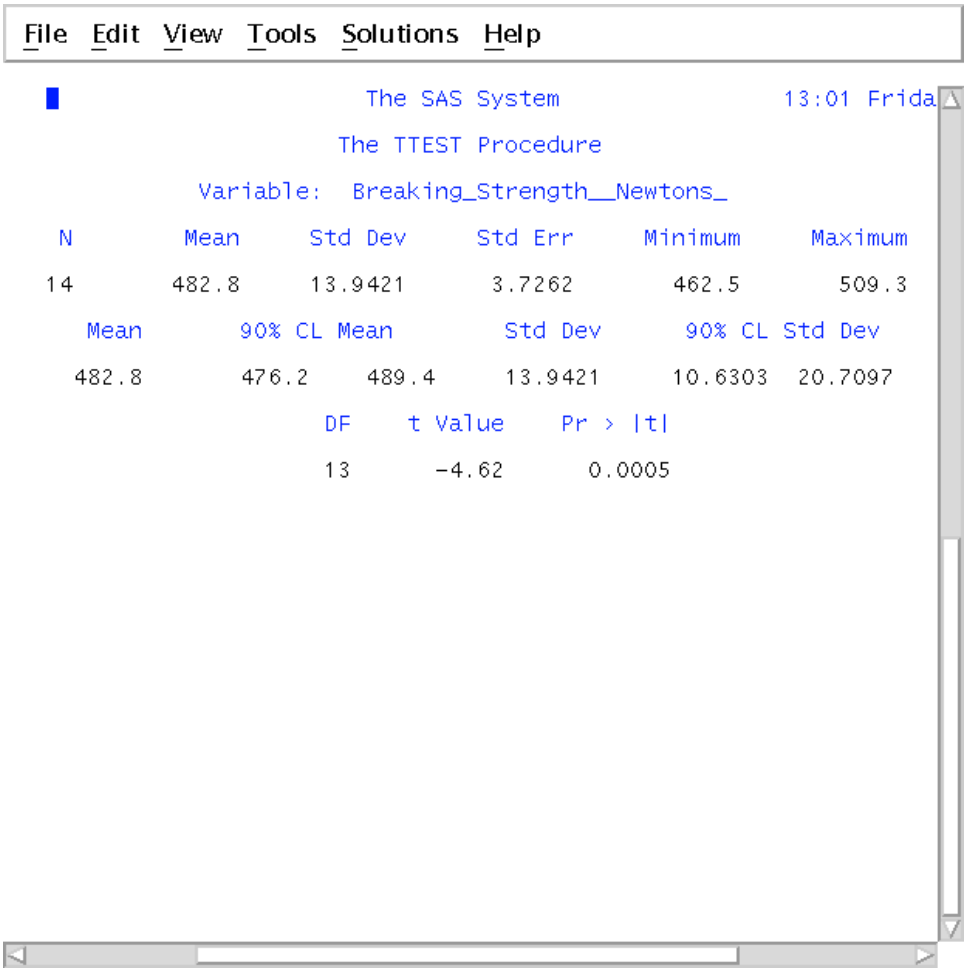


Figure 5.10. Results of our basic t test after specifying $H_0 : \mu_0 = 500$ and $\alpha = .1$

5.4 Tests of proportion

Consider a random sample of independent observations of Bernoulli random variables: $X_i \sim BIN(1,p)$. With $\hat{p} = \sum_{i=1}^{n} X_i/n$ the Central Limit Theorem ensures that

$$\frac{\hat{p}-p}{\sqrt{p(1-p)/n}} \rightarrow^{d} Z \sim N(0,1).$$

Thus for large n,

$$P\left(-z_{1-\alpha/2} < \frac{\hat{p}-p}{\sqrt{p(1-p)/n}} < z_{1-\alpha/2}\right) \approx 1-\alpha. \tag{5.4}$$

Focus on the function

$$f(p) = \frac{\hat{p}-p}{\sqrt{p(1-p)/n}}$$

on the region $\{p : 0 < p < 1\}$. Notice that

$$\lim_{p\to 0} f(p) = \infty \text{ and } \lim_{p\to 1} f(p) = -\infty.$$

It can also be shown that for every $\hat{p} \in (0,1)$, and for every sample size n, that f is a strictly decreasing function of p. The graph of f is shown for select values of p in Figure 5.11 and Figure 5.12. The confidence interval can be visualized as shown in Figure 5.13. It remains to solve for the upper and lower bounds. The lower bound is the solution of

$$\hat{p}-p = z_{1-\alpha/2}\sqrt{p(1-p)/n}. \tag{5.5}$$

The upper bound is the solution of

$$\hat{p}-p = -z_{1-\alpha/2}\sqrt{p(1-p)/n}. \tag{5.6}$$

Squaring both equations results in

$$p^2 - 2\hat{p}p + \hat{p}^2 = \frac{z_{1-\alpha/2}^2}{n}p(1-p) \tag{5.7}$$

and after rearranging we see that the two bounds are the two solutions of

$$(1+\frac{z_{1-\alpha/2}}{n})p^2 - (2\hat{p} + \frac{z_{1-\alpha/2}^2}{n})p + \hat{p}^2 = 0 \tag{5.8}$$

These can be obtained through the use of the quadratic equation and the result is

$$p = \frac{\hat{p} + \frac{z_{1-\alpha/2}^2}{2n} \pm z_{1-\alpha/2}\sqrt{\hat{p}(1-\hat{p})/n + \frac{z_{1-\alpha/2}^2}{4n^2}}}{1+\frac{z_{1-\alpha/2}^2}{n}}. \tag{5.9}$$

These two values of p are the bounds to our $(1-\alpha)$% confidence interval. This confidence interval is referred to as the Wilson confidence interval.

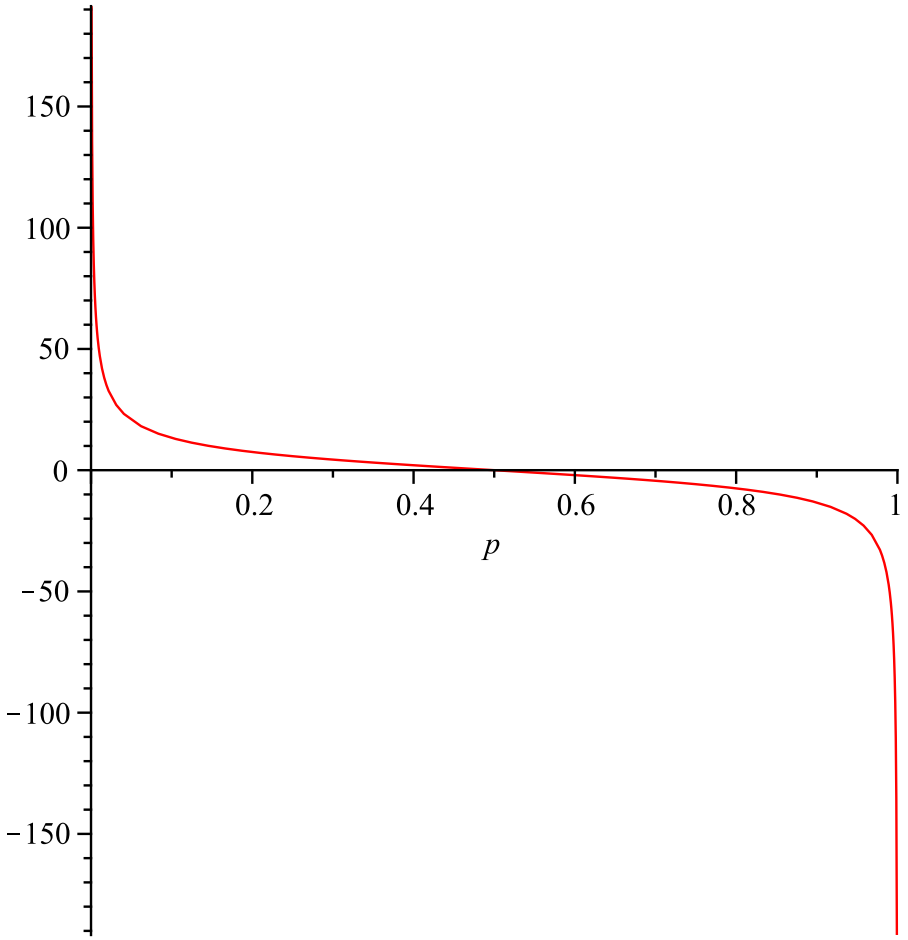


Figure 5.11. A graph of $f(p) = \frac{\hat{p}-p}{\sqrt{p(1-p)/n}}$ with $\hat{p} = .5$ and $n = 10$

Remember that this is an approximate (due to the Central Limit Theorem, see (5.4)) confidence interval. An additional approximation can also be made that avoids the above, tedious calculation. Instead of solving

$$-z_{1-\alpha/2} < \frac{\hat{p}-p}{\sqrt{p(1-p)/n}} < z_{1-\alpha/2}$$

for p we first substitute $\hat{p}$ for p in the denominator and then solve

$$-z_{1-\alpha/2} < \frac{\hat{p}-p}{\sqrt{\hat{p}(1-\hat{p})/n}} < z_{1-\alpha/2}$$

for p. This modifies 5.4 and the result is

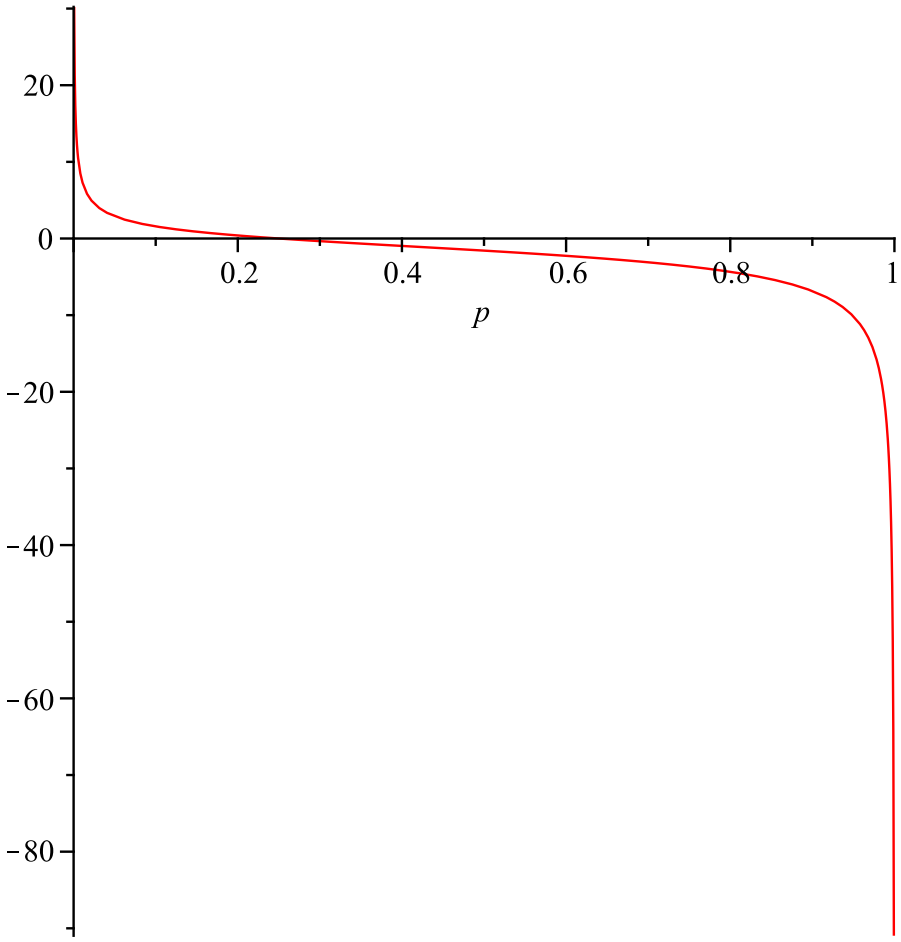


Figure 5.12. A graph of $f(p) = \frac{\hat{p}-p}{\sqrt{p(1-p)/n}}$ with $\hat{p} = .25$ and $n = 10$

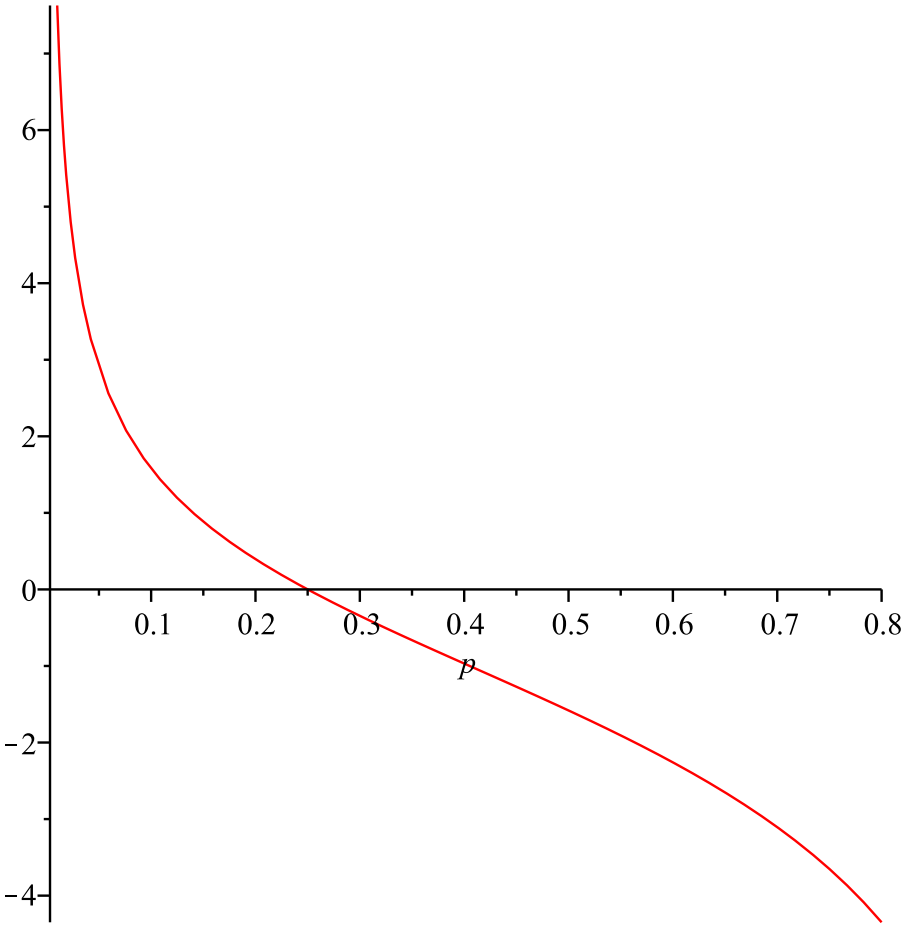


Figure 5.13. After zooming in on the graph of $f(p) = \frac{\hat{p}-p}{\sqrt{p(1-p)/n}}$ with $\hat{p} = .25$ and $n = 10$, we can visualize our confidence interval for p as $(f^{-1}(z_{1-\alpha/2}), f^{-1}(-z_{1-\alpha/2}))$. Note that this interval is not symmetric about $\hat{p}$.

$$
\begin{aligned}
&P\left(\hat{p} - z_{1-\alpha/2}\sqrt{\hat{p}(1-\hat{p})/n} < p < \hat{p} + z_{1-\alpha/2}\sqrt{\hat{p}(1-\hat{p})/n}\right) \\
&= P\left(-z_{1-\alpha/2} < \frac{\hat{p}-p}{\sqrt{\hat{p}(1-\hat{p})/n}} < z_{1-\alpha/2}\right) \\
&\approx P\left(-z_{1-\alpha/2} < \frac{\hat{p}-p}{\sqrt{p(1-p)/n}} < z_{1-\alpha/2}\right) \approx 1-\alpha.
\end{aligned}
$$

This confidence interval is known as the Agresti-Coull confidence interval. SAS can compute both the Agresti–Coull confidence interval and the Wilson confidence interval (and others), and also conduct hypothesis tests, for proportions. After first importing the data we can type the following commands.

```
PROC FREQ DATA=Work.Birthweight;
   TABLES Low / BINOMIAL(AC WILSON) ALPHA=.05;
RUN;
```

The output is shown in Figure 5.14.

Exercise 5.4. *In 1986 data was collected at the Baystate Medical Center in Springfield Massachusetts. For each of 189 different births it was recorded whether the baby was low weight (less than 2.5 kg; 1 for yes, 0 for no). This data can be accessed in the BirthWeight dataset. Load this dataset into SAS and construct both the Agresti-Coull confidence interval and the Wilson confidence interval for the population proportion of low-birth weight babies. Set* $\alpha = .10$ *so as to produce* 90% *confidence intervals. You may treat the sample as a simple random sample of independent observations.*

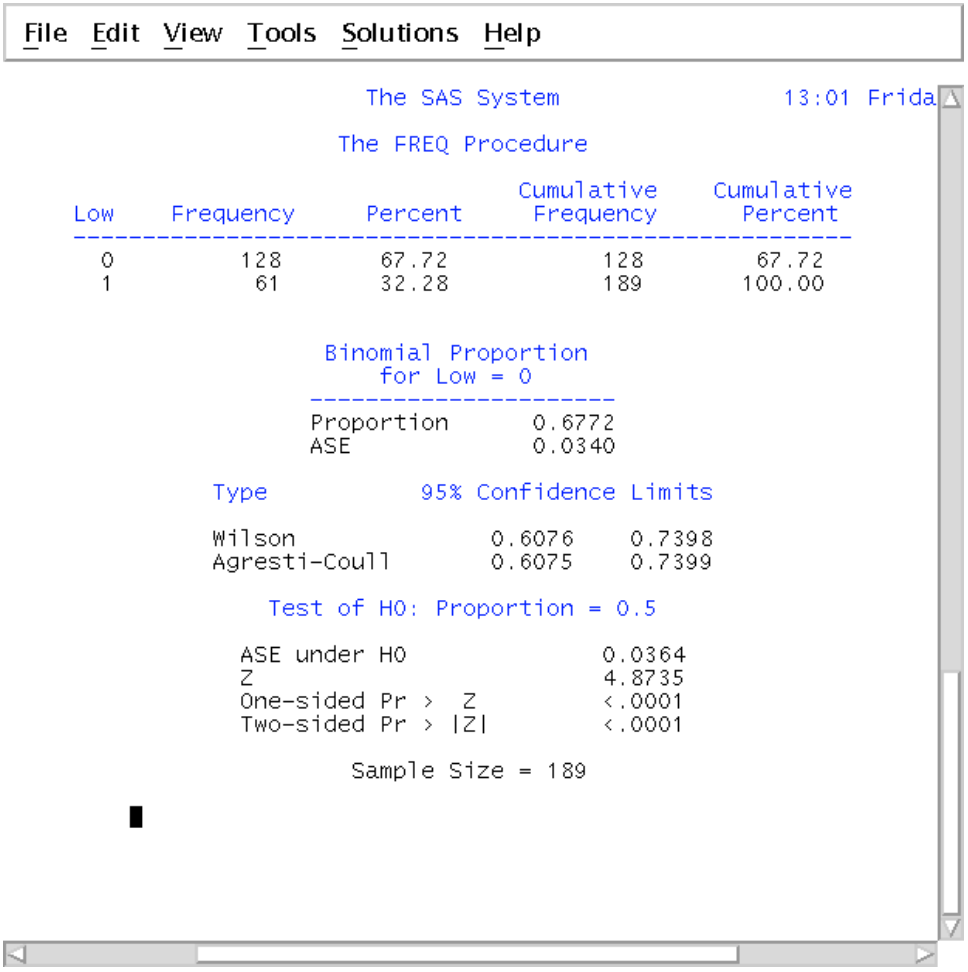


Figure 5.14. SAS PROC FREQ output. Notice the Wilson and Agresti-Coull confidence intervals in the middle.

CHAPTER 6

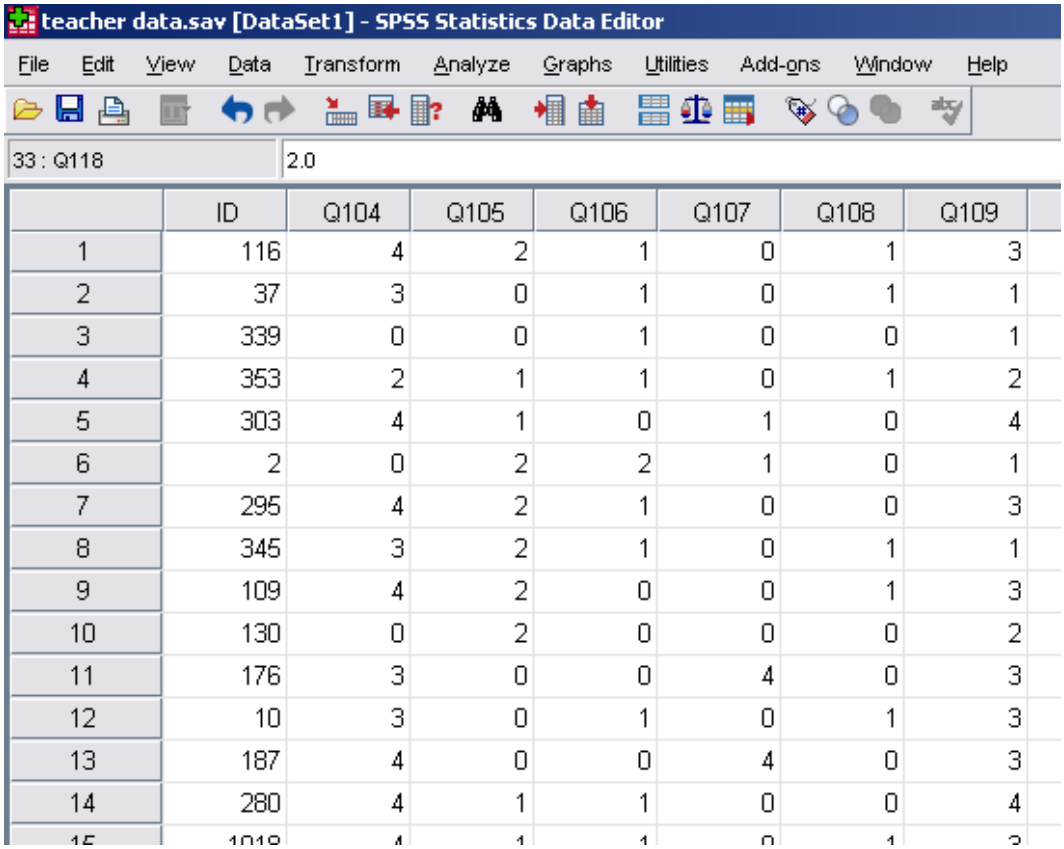
COMPARING MEANS WITH SPSS

In this chapter we will learn how to work with the computer program SPSS. SPSS is short for "Statistical Package for the Social Sciences". As of October 1st, 2001, when SPSS was acquired by IBM it has gone by the name IBM SPSS. We will use the abbreviation SPSS here. SPSS was designed to be used in the social sciences, however, SPSS is easily capable of carrying out basic statistical analyses, regardless of the type of data. The chief advantage of SPSS is perhaps its user friendly format. Throughout this chapter we will emphasize how the user of SPSS can 'point and click' their way through a statistical analysis. We will start by posing three sample problems before discussing the mathematical theory of ANOVA which we will then keep in mind while solving these problems using SPSS. This chapter provides a nice opportunity to see both the students-t statistic and the F statistic in action.

6.1 Sample problems

We will be analyzing a data set called 'TeacherData.sav' which contains data from 337 Utah teachers. It is straightforward to point and click ones way to opening the dataset within SPSS. Likewise, simply clicking on appropriate tabs will toggle between SPSS's data view and variable view as we will see shortly.

The dataset comes from a training session and contains many variables worth of data for each teacher who attended the training session. Before the training they took a test and their scores were recorded under 'prior to training'. After the training they took another test and their scores were recorded under 'after training'. In addition, information relevant to their teaching, and test scores, was recorded. Examples include gender, age, highest degree obtained, number of years of teaching experience, etc. In all there are 28 variables associated with each teacher. This results in a data set that has 337 rows and 28 columns. Figure 6.1 shows rows 1–15 in SPSS "Data View". We can also select "Variable View" to see how each of the variables (columns) is defined. Much goes into each variable's definition. The important topics are highlighted in Figure 6.2. The 28 variables are indexed in the left–most column. Each variable has a Name and a Label.



	ID	Q104	Q105	Q106	Q107	Q108	Q109
1	116	4	2	1	0	1	3
2	37	3	0	1	0	1	1
3	339	0	0	1	0	0	1
4	353	2	1	1	0	1	2
5	303	4	1	0	1	0	4
6	2	0	2	2	1	0	1
7	295	4	2	1	0	0	3
8	345	3	2	1	0	1	1
9	109	4	2	0	0	1	3
10	130	0	2	0	0	0	2
11	176	3	0	0	4	0	3
12	10	3	0	1	0	1	3
13	187	4	0	0	4	0	3
14	280	4	1	1	0	0	4

Figure 6.1. SPSS Data View for 'TeacherData', Columns 1 through 14

The Label provides meaning and clarity. For example, question 106 was entered under the name Q106, but not until we see the Label (Reading First school) do we know what this variable is all about. Teachers were asked whether they work at a Reading First school. If they answered 'yes' this was coded and entered as a '0', if they answered 'no' this was coded and entered as a '1', as outlined in the Values column. The statistician who analyzes the data will do so by viewing Labels (such as Reading First school) and answers (such as 'yes' or 'no') whereas behind the scenes SPSS will be working with the numbers. To see the entire coding scheme, click on one of the boxes under Values, such as that for Level of preparedness. It's a good idea to spend some time in Variable View in order to become familiar with the meanings of all the numbers that one sees in Data View. Regardless of the view, there is a menu bar with headings such as Analyze, and Graphs. This is where the action happens, as will soon be illustrated (see sections 6.3 and 6.4).

For now we look at some typical questions that might need answering. Keep in mind that we are considering this data set to be a random sample of Utah teachers. The idea being that we can use the data to make general claims about Utah teachers. Thus the following questions regard

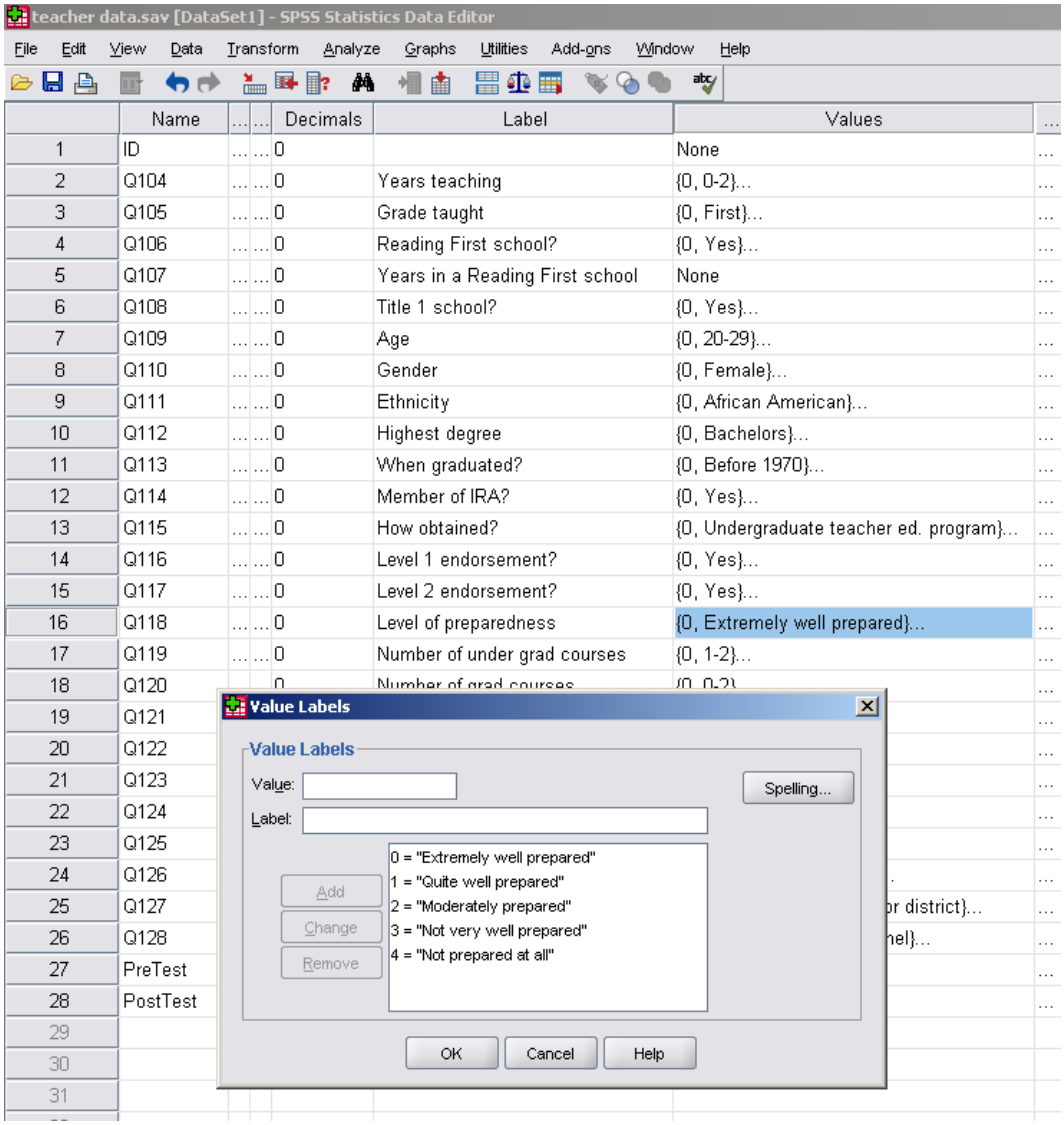


Figure 6.2. SPSS Variable View with the *Value Labels* box in view

Utah teachers.

6.1.1 Comparing two means: T-tests

The following questions all involve the comparison of two means.

1. Do female teachers score higher on the pre test than male teachers?

2. Do teachers from title-one schools score higher on the pre test?

3. Do teachers with a level 2 endorsement score higher on the pre test?

4. Do teachers from title-one schools learn more during training than the other teachers?

Each of these questions can be answered by conducting t-tests (see section 6.3).

6.1.2 One-way ANOVA

The following questions each involve the comparison of three (or more) means.

1. Do 1st grade teachers, 2nd grade teachers, and 3rd grade teachers all perform comparably on the pre test?

2. All teachers presumably belong to one of four 'level of preparedness' groups. Does each group score about the same (on average) on the pre test?

3. Did each group improve (as measured by post test minus pre test) about the same on average, during the training?

Each of these questions can be answered using One-way ANOVA (see section 6.4)

Exercise 6.1. *Use SPSS to open the 'TeacherData.sav' dataset. Examine both the data view and the variable view. Make sure to understand the meaning behind each variable name as well as the different values that each variable can take.*

6.2 Mathematical theory

We will answer questions, such as those posed in the previous section, based on the mathematical theory as discussed here. We will be comparing population means based on the information obtained via samples. We start with the following assumptions.

- Independent random samples have been taken from each population.
- The populations are normal.
- The population variances are equal.

6.2.1 T-test theory

We have already dealt with single–sample T-tests (see Theorem 5.1). Our single sample provided the information to test whether the population mean was or was not equal to some specified value. Here we deal with two populations and introduce two related two–sample T-tests that are computed automatically by SPSS when comparing population means.

The first is based on the following theorem and assumes equal variances.

Theorem 6.1. *If $X_1, ..., X_{n_1}$ is a random sample from $N(\mu_1, \sigma^2)$ and $Y_1, ..., Y_{n_2}$ is an independent random sample from $N(\mu_2, \sigma^2)$ then with the pooled standard deviation defined as*

$$S_p = \sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}}$$

we have

$$\frac{(\bar{X}-\bar{Y}) - (\mu_1-\mu_2)}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} =^d T(n_1+n_2-2).$$

Proof. See Bain and Engelhardt [4], Chapter 8. □

Corollary 6.2. *(Two–sample T-test) Assuming two independent samples (sizes n_1 and n_2) from normal populations with equal variances, a two-sided, size α test of $H_0 : \mu_1 = \mu_2$, vs $H_a : \mu_1 \neq \mu_2$, uses the statistic*

$$\frac{\bar{X}-\bar{Y}}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} =^d T(n_1+n_2-2)$$

and rejects H_0 if

$$\left|\frac{\bar{x}-\bar{y}}{s_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}\right| > t_{1-\alpha/2}(n_1+n_2-2).$$

6.2.2 One-way ANOVA theory

When there are k different populations, rather than running $k!$ pairwise t-tests, we will run a single test that allows us to check whether all the population means are the same. This test is the simplest version of Analysis of Variance, or ANOVA. Keep in mind that this title refers to the analysis, which deals with variances. The end goal of the test is to detect differences in the different population means. The test will utilize the following statistic:

$$\frac{\sum_{j=1}^{n_j} n_j(\bar{X}_j - \bar{\bar{X}})^2/(k-1)}{\sum_{j=1}^{k}\sum_{i=1}^{n_j}(X_{ij}-\bar{X}_j)^2/(N-k)}.$$

$\bar{X}_j$ is the mean of the jth sample (size n_j).

$$\bar{\bar{X}} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n_j} X_{ij}}{\sum_{j=1}^{k} n_j} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n_j} X_{ij}}{N}$$

is the grand mean of all $N = \sum_{j=1}^{k} n_j$ observations. The statistic can be interpreted as follows. The numerator reflects how spread out the different samples are from each other. The denominator moderates by taking within-sample variance into account. Thus our statistic is small under the null and large under the alternative. In fact, under the null, the statistic follows an F distribution.

Theorem 6.3. *For k normal populations (indexed with j), each with the same mean and variance, the following statistic follows an F distribution:*

$$\frac{\sum_{j=1}^{n_j} n_j(\bar{X}_j - \bar{\bar{X}})^2/(k-1)}{\sum_{j=1}^{k}\sum_{i=1}^{n_j}(X_{ij} - \bar{X}_j)^2/(N-k)} =^d F(k-1, N-k).$$

Proof. See Bain and Engelhardt ([4]), Theorem 12.8.1. □

This leads us to the following.

Corollary 6.4. *(One-way Anova Hypothesis Test) Assuming k independent samples from normal populations with equal variances, a size α test of $H_0 : \mu_1 = ... = \mu_k$ uses the statistic*

$$\frac{\sum_{j=1}^{k} n_j(\bar{X}_j - \bar{\bar{X}})^2/(k-1)}{\sum_{j=1}^{k}\sum_{i=1}^{n_j}(X_{ij} - \bar{X}_j)^2/(N-k)} =^d F(k-1, N-k)$$

and rejects H_0 if

$$\frac{\sum_{j=1}^{k} n_j(\bar{x}_j - \bar{\bar{x}})^2/(k-1)}{\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij} - \bar{x}_j)^2/(N-k)} > f_{1-\alpha}(k-1, N-k).$$

6.2.3 When the model assumptions fail

The theory from Section 6.2 holds under the assumptions of normality and equal variances.

In the case of the two-sample T-test for example, we can check the equal-variances assumption by examining the statistic

$$\frac{S_1^2}{S_2^2}. \tag{6.1}$$

We know [4] that upon assuming normality that the distribution of the sample standard deviation can be related to a chi-squared distribution:

$$(n-1)S^2/\sigma^2 =^d \chi^2(n-1)$$

so that for independent samples

$$\frac{S_1^2/\sigma_1^2}{S_2^2/\sigma_2^2} =^d \frac{\chi^2(n_1-1)/(n_1-1)}{\chi^2(n_2-1)/(n_2-1)} = F(n_1-1, n_2-1).$$

Under the null hypothesis of equal variances, $\frac{\sigma_1^2}{\sigma_2^2} = 1$ and we see that our statistic from (6.1) is distributed as an F:

$$\frac{S_1^2}{S_2^2} =^d F(n_1-1, n_2-1).$$

This is the basis for an F-test of equal variances.

To test the hypothesis of equal variances across $k > 2$ populations, we could use Bartlett's test. With k samples, each having sample size N_i, $N = \sum_{i=1}^{k} N_i$, and S_p^2 denoting the pooled, sample variance,

$$S_p^2 = \frac{\sum_{i=1}^{k} S_i^2}{\sum_{i=1}^{k}(N_i - 1)},$$

Bartlett's test uses the following statistic:

$$\frac{(N-k)\log\left(S_p^2 \sum_{i=1}^{k}(N_i - 1)\log(S_i^2)\right)}{(1 + (k-1)/3)\left(\sum_{i=1}^{k}(1/(N_i - 1)) - 1/(N-k)\right)}.$$

Under the assumptions of normality and equal variances across groups (homoscedasticity), the test statistic follows a $\chi^2(k-1)$ distribution. This can be used to test the null hypothesis of equal variances across groups, but as with our previously mentioned F-test, this test will again not be robust against departures from normal data. For more reading on Bartlett's test see a page from the National Institute of Standards and Technology's Engineering Statistics Handbook ([13]).

A more robust procedure for testing equality of variances across multiple groups exists and is called Levene's test. This test will be run automatically whenever conducting a T-test or one-way ANOVA test within SPSS. We summarize the basics of the test here.

Levene's test is based on the following statistic:

$$\frac{(N-k)}{(k-1)} \frac{\sum_{i=1}^{k} N_i (\bar{Z}_i - \bar{\bar{Z}})^2}{\sum_{i=1}^{k}\sum_{j=1}^{N_i}(Z_{ij} - \bar{Z}_i)^2}, \tag{6.2}$$

where N is the total number of observations and each N_i is the number of observations in the ith sample. The Z variables are functions of the (to-be-observed) X variables according to,

- $Z_{ij} = |X_{ij} - \bar{X}_i|$ (the absolute deviation of a measurement from its group mean).
- $\bar{\bar{Z}} = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{N_i} Z_{ij}$.
- $\bar{Z}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} Z_{ij}$.

Upon examination of the second fraction in 6.2 we see that as we move away from homoscedastic samples, the numerator grows. The growth rate depends on the spread of the samples however; the denominator of the second fraction takes this into account. If we assume normality across the groups then the statistic is distributed as an $F(k-1, N-k)$. Levene's test then rejects homoscedasticity for large values of the computed statistic. Note also that Levene's test is more robust against departures from normality than the previously mentioned Bartlett's test ([14]).

It should be noted that when we defined Z_{ij} we used the sample mean. Had we used the median the result would be the Brown–Forsythe test. The trimmed mean can also be used. Distributional assumptions will determine which version is most appropriate. We will stick to SPSS's default settings and deal with Levene's test only. For more details consult the National Institute of Standards and Technology's Engineering Statistics Handbook ([14]).

To effectively utilize Levene's test within SPSS one need only keep in mind that the null hypothesis states that all population variances are equal, and that a significant p-value can be interpreted as reason to reject this equal-variances assumption.

If we have reason to believe that the variances across different populations are all equal, and Levene's test failes to reject this assumption, it is then possible, under the assumption of normality, to use the two-sample t-test or the one-way ANOVA test to compare population means, as theoretically discussed in Sections 6.2.2 and 6.2.1.

If, however, we have reason to doubt the equal variances assumption, such as perhaps a significant rejection coming from Levene's test, we should then turn to alternative methods.

SPSS does not automatically present an alternative test, to be used in place of one-way ANOVA, in case the equal variances assumption is rejected. In such a scenerio we should perhaps turn to simulation as a means to compare $k > 2$ population means. For an introduction to simulation using R see Chapter 3.

SPSS does however present an alternative test, the unequal variances T-test, that can be used to compare 2 population means, even when the population variances are thought to be unequal.

This test can be contrasted with the equal variances T-test as discussed in Section 6.2.1 where we used the statistic

$$\frac{(\bar{X}-\bar{Y})-(\mu_1-\mu_2)}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}$$

and found it to follow a T distribution. The proof of this fact used the equal variances assumption though. Without equal variances the statistic

$$\frac{(\bar{X}-\bar{Y})-(\mu_1-\mu_2)}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}$$

is no longer distributed as a T distribution.

Thus we modify our approach. A relevant statistic to consider is still

$$(\bar{X}-\bar{Y})-(\mu_1-\mu_2).$$

After dividing by its standard deviation we see that it is distributed normally:

$$\frac{(\bar{X}-\bar{Y})-(\mu_1-\mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}} =^d N(0,1).$$

However, the theoretical quantities σ_1^2 and σ_2^2 are unknown.

We met the same problem when assuming $\sigma_1^2=\sigma_2^2=\sigma^2$ and found that upon substituting S_p^2 for σ^2 in

$$\frac{(\bar{X}-\bar{Y})-(\mu_1-\mu_2)}{\sqrt{\frac{\sigma^2}{n_1}+\frac{\sigma^2}{n_2}}} =^d N(0,1)$$

that the result was a T:

$$\frac{(\bar{X}-\bar{Y})-(\mu_1-\mu_2)}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} =^d T(n_1+n_2-2).$$

Without the unequal variances assumption we can instead make two substitutions. If we substitute S_1^2 for σ_1^2 and S_2^2 for σ_2^2 in

$$\frac{(\bar{X}-\bar{Y})-(\mu_1-\mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}} =^d N(0,1)$$

the result is the statistic

$$T=\frac{(\bar{X}-\bar{Y})-(\mu_1-\mu_2)}{\sqrt{\frac{S_1^2}{n_1}+\frac{S_2^2}{n_2}}}. \tag{6.3}$$

We have labeled this statistic with a T because it is approximately distributed as a T distribution, but not exactly. Furthermore, if we are to treat it as a T then we must know its degrees of freedom. We approximate the degrees of freedom by considering variances as follows. Note that T refers to the statistic from (6.3) while $T(\nu)$ is the theoretically defined T distribution with ν degrees of freedom. We use $\approx^d$ to indicate approximate equality in distribution. We have assumed

$$T=\frac{(\bar{X}-\bar{Y})-(\mu_1-\mu_2)}{\sqrt{\frac{S_1^2}{n_1}+\frac{S_2^2}{n_2}}} \approx^d T(\nu)=\frac{N(0,1)}{\sqrt{\chi^2(\nu)/\nu}}. \tag{6.4}$$

Assuming normality and independent samples, we know [4] that the four random quantities, $\bar{X}$, $\bar{Y}$, S_1^2 and S_2^2 are all independent and thus that

$$\frac{\frac{S_1^2}{n_1}+\frac{S_2^2}{n_2}}{\frac{\sigma_x^2}{n_1}+\frac{\sigma_y^2}{n_2}} \approx^d \frac{\chi^2(\nu)}{\nu}. \tag{6.5}$$

We next compute the variance of both sides of (6.5). We know in general [4] that

$$\mathrm{Var}(\chi^2(\nu))=2\nu. \tag{6.6}$$

This allows us to compute the variance of the right hand side of (6.5):

$$\text{Var}\left(\frac{\chi^2(\nu)}{\nu}\right) = \frac{2\nu}{\nu^2} = \frac{2}{\nu}. \tag{6.7}$$

Again, since we are assuming normal data,

$$S^2 =^d \chi^2(n-1)\sigma^2/(n-1).$$

We use this for each sample variance, and by again appealing to equation (6.6) we can compute the variance of the numerator from the left hand side of (6.5).

$$\begin{aligned}
\text{Var}\left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}\right) &= \text{Var}\left(\frac{\chi^2(\text{n}_1-1)\sigma_1^2}{\text{n}_1(\text{n}_1-1)} + \frac{\chi^2(\text{n}_2-1)\sigma_2^2}{\text{n}_2(\text{n}_2-1)}\right) \\
&= \frac{\text{Var}(\chi^2(\text{n}_1-1))\sigma_1^4}{n_1^2(n_1-1)^2} + \frac{\text{Var}(\chi^2(\text{n}_2-1))\sigma_2^4}{n_2^2(n_2-1)^2} \\
&= \frac{2(n_1-1)\sigma_1^4}{n_1^2(n_1-1)^2} + \frac{2(n_2-1)\sigma_2^4}{n_2^2(n_2-1)^2} \\
&= 2\left(\frac{\sigma_1^4}{(n_1-1)n_1^2} + \frac{\sigma_2^4}{(n_2-1)n_2}\right).
\end{aligned} \tag{6.8}$$

Thus,

$$\text{Var}\left(\frac{\frac{\text{S}_1^2}{\text{n}_1} + \frac{\text{S}_2^2}{\text{n}_2}}{\frac{\sigma_1^2}{\text{n}_1} + \frac{\sigma_2^2}{\text{n}_2}}\right) = \frac{2\left(\frac{\sigma_1^4}{(\text{n}_1-1)\text{n}_1^2} + \frac{\sigma_2^4}{(\text{n}_2-1)\text{n}_2}\right)}{\left(\frac{\sigma_1^2}{\text{n}_1} + \frac{\sigma_2^2}{\text{n}_2}\right)^2}. \tag{6.9}$$

Assuming now that our distributional approximation in (6.5) is strong enough that the variances for each distribution are approximately equal, we then have (6.5), (6.7) and (6.9) implying

$$\frac{2}{\nu} \approx \frac{2\left(\frac{\sigma_1^4}{(n_1-1)n_1^2} + \frac{\sigma_2^4}{(n_2-1)n_2}\right)}{\left(\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right)^2} \iff \nu \approx \frac{\left(\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right)^2}{\left(\frac{\sigma_1^4}{(n_1-1)n_1^2} + \frac{\sigma_2^4}{(n_2-1)n_2}\right)}. \tag{6.10}$$

The quantity

$$\frac{\left(\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right)^2}{\left(\frac{\sigma_1^4}{(n_1-1)n_1^2} + \frac{\sigma_2^4}{(n_2-1)n_2}\right)}$$

can be estimated with

$$\frac{(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2})^2}{\frac{S_1^4}{(n_1-1)n_1^2} + \frac{S_2^4}{(n_2-1)n_2}}.$$

This is precisely what we need: an estimate for the approximating degrees of freedom. We can now describe a procedure for comparing two population means, even when the variances are unequal.

We know that

$$T = \frac{(\bar{X} - \bar{Y}) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \approx^d T(\nu) \tag{6.11}$$

with ν given by

$$\frac{(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2})^2}{\frac{S_1^4}{(n_1-1)n_1^2} + \frac{S_2^4}{(n_2-1)n_2}}. \tag{6.12}$$

or the closest natural number. Our two-tailed test of $H_0 : \mu_1 = \mu_2$ vs $H_a : \mu_1 \neq \mu_2$ then consists of examining

$$t = \frac{(\bar{x} - \bar{y})}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

and rejecting H_0 if

$$|t| > c,$$

where c is determined from the approximating $T(\nu)$ distribution as described in (6.11) and (6.12).

While we have provided some justification for the above procedure it should be noted that the procedure is largely empirically based. Studies have been done and verified that the procedure works well in practice. Even without normal data, if either the sample distributions and sizes are roughly equal, with size greater than five, or even with clearly skewed samples of different distributions and sizes, as long as the sum of the sizes is greater than forty, then the procedure is reliable [7].

In the following section we will see how SPSS carries out the procedure.

Exercise 6.2. *Gain familiarity with the graphical abilities of SPSS by clicking on Graphs and then Chart Builder. Explore any variables of interest from the TeacherData dataset by producing histograms.*

6.3 T-tests with SPSS

Here we use SPSS to answer the questions as posed in section 6.1.

Assuming that our samples can be treated as independent random samples from the population of Utah teachers we proceed to plot pre test-score histograms for female and male teachers.

To do this we first open *Chart Builder* by clicking on *Graphs* and then *Chart Builder*. Then we click on *Histogram*, which allows us to choose an appropriate histogram. Choose the *Population Pyramid* histogram and drag it into the top window. Next we position the appropriate variables. Under *Variables* click on *Gender* and drag it to the right, placing it within the *Split Variable* tab.

Then we click on *prior to training* (which stands for the pretest score) and drag it to the right as well, placing it within the *Distribution Variable* tab. Click *OK* to execute. The histogram should appear in the output window as shown in Figure 6.3. Note that the data have been split into three groups. The third contains those observations lacking a male or female designation, perhaps due to error or other reasons. For our purposes these observations can be ignored. Upon inspecting the female and male histograms we find no obvious reasons for rejecting the normality or equal variance assumptions. We thus proceed to the t-test where Levenes test of equal variances will be conducted automatically.

To run the independent-samples t-test click on *Analyze*, *Compare Means*, and then *Independent Samples T-Test*. A new window will pop up. Choose *Gender* as the *Grouping Variable*. SPSS allows us the flexibility to define our groups (based on the observations of the grouping variable), although in this instance we simply specify the obvious: clicking on *Define Groups* we specify '0' for group 1 and '1' for group 2. This selects 354 observations of females to be the first group (sample) and 18 observations of males to be the second group (sample). Finally, we choose *prior to training* (pre test scores) as the *Test Variable* before hitting *Continue* and

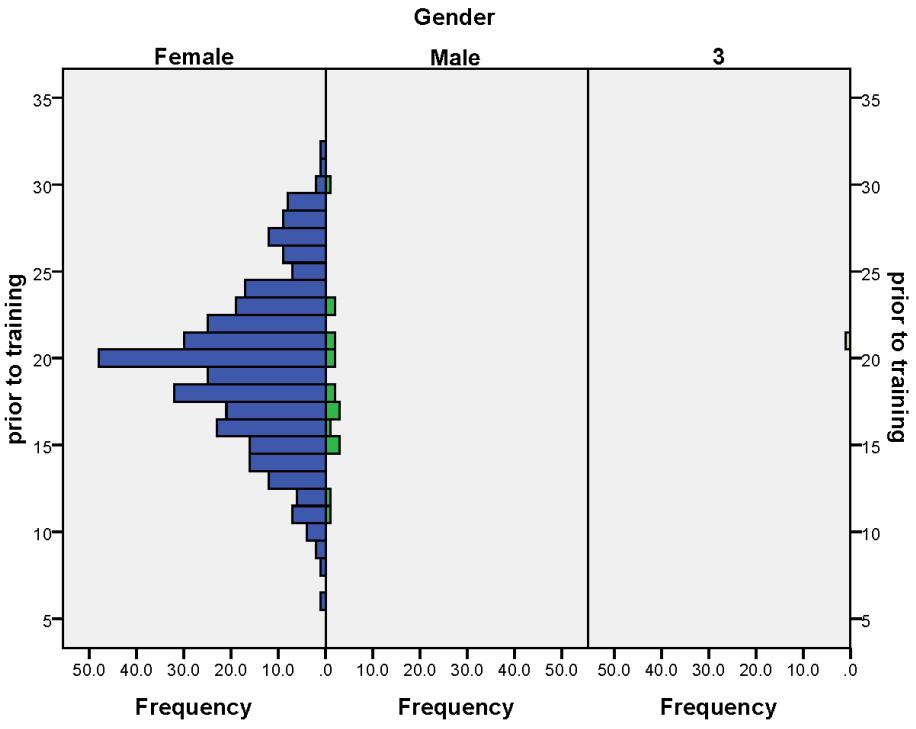


Figure 6.3. Histograms grouped by 'Gender' showing the sample distribution of pre test-scores

then *OK* to conduct the test. The results will appear in the output window as shown in Figure 6.4. The first thing to check is Levene's test for equality of variances. The p-value is displayed under 'Sig.' and as expected it is not significant in this case. Thus, both tests may provide useful information. The results for both tests are displayed in the lowest box and we see that neither t value is deemed significant. Thus we have reached an answer to our question. While females on average scored higher (19.61 vs 18.28) the T-test results ($t(370) = 1.196$, $p = .232$) failed to reach statistical significance, even at the $\alpha = .1$ level. Thus we conclude that the apparent difference in scores is perhaps simply due to the element of chance in the random sampling.

Another question we might need to answer is the following.

2. Do teachers with a level 2 endorsement score higher on the pre test?

Again, we start with a histogram as shown in Figure 6.5. The figure alerts us to the fact that both the normality and the equal variances assumptions might not hold. However, with such large sample sizes, we expect robustness from the 'equal variances not assumed' T procedure.

Group Statistics

	Gender	N	Mean	Std. Deviation	Std. Error Mean
prior to training	Female	354	19.61	4.610	.245
	Male	18	18.28	4.456	1.050

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means	
		F	Sig.	t	df
prior to training	Equal variances assumed	.148	.700	1.198	370
	Equal variances not assumed			1.235	18.898

Independent Samples Test

		t-test for Equality of Means		
		Sig. (2-tailed)	Mean Difference	Std. Error Difference
prior to training	Equal variances assumed	.232	1.332	1.112
	Equal variances not assumed	.232	1.332	1.079

Figure 6.4. Results of the t-test for 'prior to training' grouped by 'Gender'

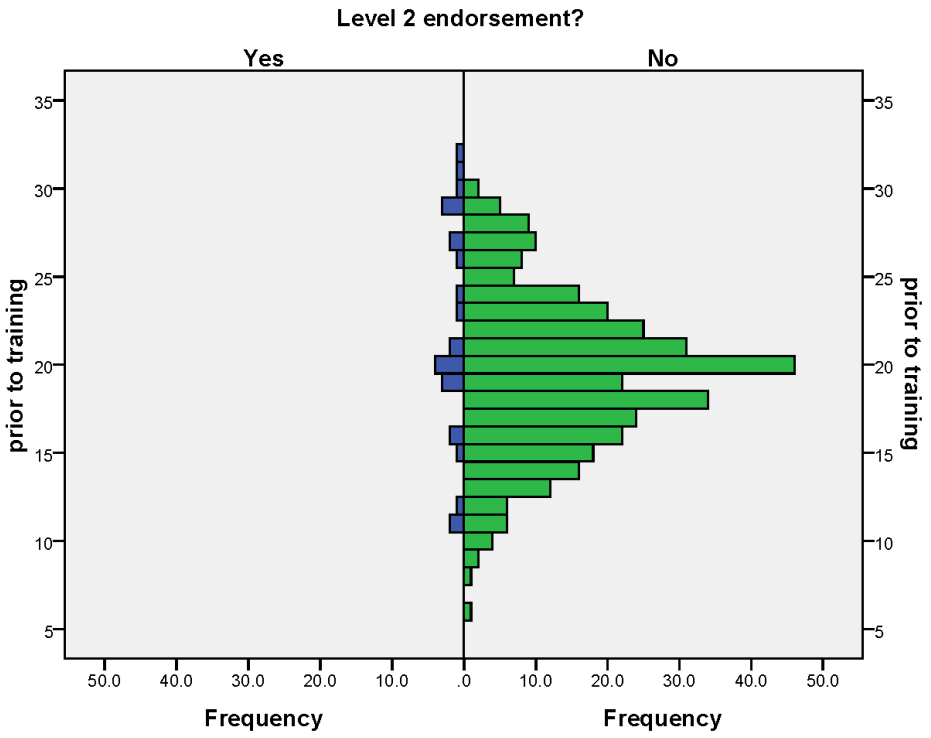


Figure 6.5. Histograms grouped by 'Level 2 endorsement' showing the sample distribution of pre test scores

Running an independent samples T-test results in a printout as shown in Figure 6.6. Levene's test rejects equal variances so we focus directly on the 'equal variances not assumed' row of the final table. Using the standard $\alpha = .05$ level of significance we see that the results ($t(26.919) = 1.950$, $p = .062$) are in fact insignificant. Had we blindly assumed equal variances we would have reported significant results ($t(371) = 2.615$, $p = .009$). This shows the importance of questioning the standard assumptions, and also how the degrees of freedom can significantly effect p-values.

Our final question is a matched pairs problem.

3. Do teachers from title-one schools learn more during training than the other teachers?

This problem involves a slight twist. The dependent variable is no longer the pre test score. It is now the difference between the pre test score and the post test score. We treat this as an opportunity to learn how to define new variables within SPSS.

Click on *Transform* and then *Compute Variable*. Label the new *Target Variable* as *Improvement* and define it within the *Numeric Expression* box as *PostTest-PreTest*. To this end simply drag the appropriate labels (*after training* and *prior to training*) into the box. Then click *OK* and the 29th variable, 'Improvement', is now a regular part of the data.

Group Statistics

	Level 2 endorsement?	N	Mean	Std. Deviation	Std. Error Mean
prior to training	Yes	26	21.81	6.229	1.222
	No	347	19.38	4.420	.237

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means	
		F	Sig.	t	df
prior to training	Equal variances assumed	8.927	.003	2.615	371
	Equal variances not assumed			1.950	26.919

Independent Samples Test

		t-test for Equality of Means		
		Sig. (2-tailed)	Mean Difference	Std. Error Difference
prior to training	Equal variances assumed	.009	2.427	.928
	Equal variances not assumed	.062	2.427	1.244

Figure 6.6. Results of the t-test for 'prior to training' grouped by 'Level 2 endorsement'

The next step is then to make a side by side histogram as illustrated in Figure 6.7. The histogram is consistent with both the normality and equal variances assumptions. To conduct a T-test we proceed as in example **??**, but with *Title 1 school* as the *Grouping Variable* and *Improvement* as the *Test Variable*. The output is shown in Figure 6.8 Both T-tests give nearly identical results and lead us to the same conclusion. Teachers from Title 1 schools don't seem to learn any more or less than the other teachers during training sessions. But, do they score higher on the tests?

Exercise 6.3. *Do teachers from title-one schools score higher on the pre test?*

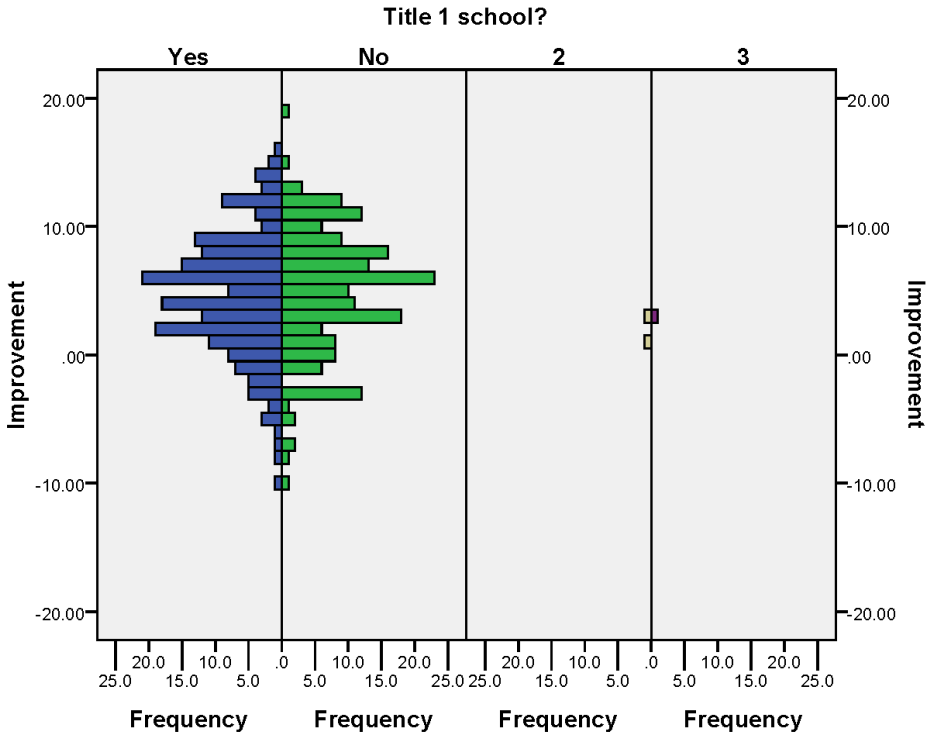


Figure 6.7. Histograms grouped by 'Title 1 school', showing the sample distribution of 'Improvement'

Group Statistics

	Title 1 school?	N	Mean	Std. Deviation	Std. Error Mean
Improvement	Yes	189	4.6455	4.84946	.35275
	No	179	4.9777	4.90695	.36676

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means	
		F	Sig.	t	df
Improvement	Equal variances assumed	.005	.943	-.653	366
	Equal variances not assumed			-.653	364.399

Independent Samples Test

		t-test for Equality of Means		
		Sig. (2-tailed)	Mean Difference	Std. Error Difference
Improvement	Equal variances assumed	.514	-.33215	.50870
	Equal variances not assumed	.514	-.33215	.50887

Figure 6.8. Results of the t-test for 'Improvement' grouped by 'Title 1 school'

6.4 One-way ANOVA with SPSS

Comparing three or more population means (running one way ANOVA) is similar to comparing two population means (running a T-test) although there is no simple alternative procedure when the equal variances assumption is shown to be false. Thus, we need not run Levene's test with the aim of deciding which T-test to use. If the groups have wildly different variances per perhaps a visual plot of the histograms side by side will serve as the best evidence against equal means. Conversely, the side by side histogram plot can provide evidence for equal variances, and since one way ANOVA is robust against deviations from normality (see Kirk [8], Norusis [5]) after obtaining a satisfactory histogram plot we can feel confident in the results of the one way ANOVA test. So, keeping the need for histograms in mind we meet our first ANOVA example.

1. Do 1st grade teachers, 2nd grade teachers, and 3rd grade teachers all perform comparably on the pre test?

The histogram can be produced as described previously: start by clicking on *Chart Builder*. The resulting output is shown in Figure 6.9. There are no obvious reasons to reject either the normality or equal variance assumptions so we proceed to test the hypothesis that for each grade level 1,2,3 the mean pre test score is the same, $H_0 : \mu_1 = \mu_2 = \mu_3$.

To run ANOVA we click on *Analyze*, *Compare Means*, and then *One-Way-ANOVA* resulting in a one-way-ANOVA box. We next move *Grade taught* into the *Factor* box and *prior to training* into the *Dependent List*. Then we click *OK* and the results appear in the output window as shown in Figure 6.10. Remember that the computed statistic

$$\frac{\sum_{j=1}^{n_j} n_j(\bar{X}_j - \bar{\bar{X}})^2/(k-1)}{\sum_{j=1}^{k}\sum_{i=1}^{n_j}(X_{ij} - \bar{X}_j)^2/(N-k)}$$

is distributed, under H_0, as $F(k-1, N-k)$. Thus, we see a box labeled with F as part of the ANOVA output. It is the ratio of the two reported 'Mean Square' values, the numerator is labeled with 'between groups' and the denominator with 'within groups'. This terminology is self-explanatory.

The 'Sum of Squares' column is slightly mysterious. It's not a coincidence that the first two numbers sum to give the third. This fact, while not needed in this chapter, can be appreciated through careful study of definitions to be found in Section 7.5.

Similarly, the degrees of freedom column is not essential, although it is useful when formally reporting the observed F value.

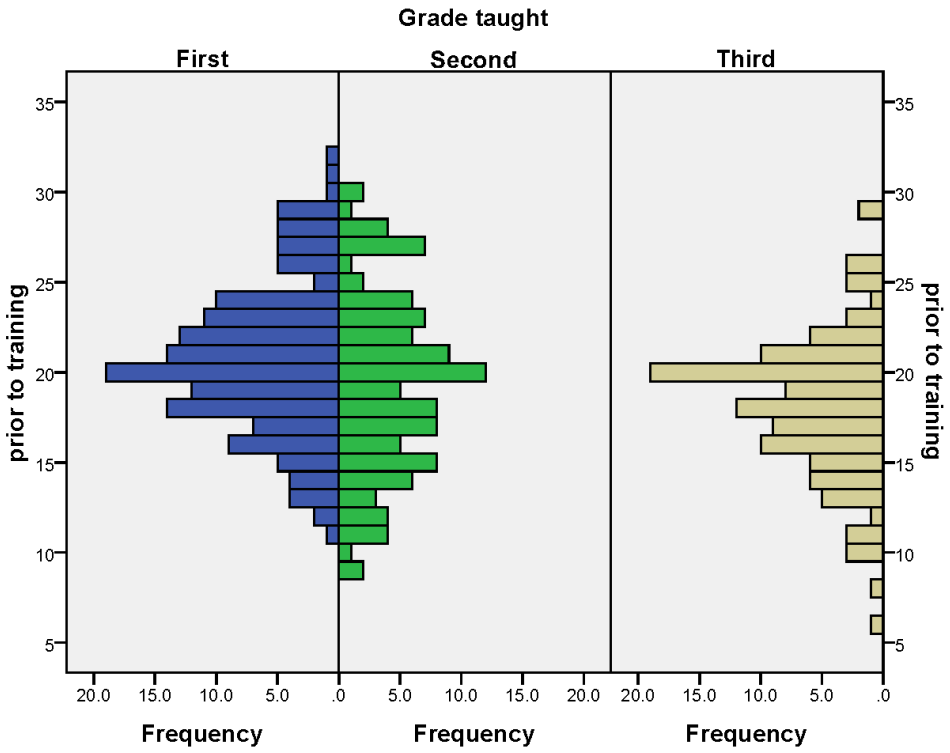


Figure 6.9. Histograms grouped by 'Grade taught', showing the sample distribution of pre test scores

ANOVA

prior to training

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	420.026	2	210.013	10.427	.000
Within Groups	7452.306	370	20.141		
Total	7872.332	372			

Figure 6.10. Resulting ANOVA table for 'prior to training' grouped by 'Grade taught' as produced by SPSS

Our focus should be on the final column - the significance column. This column contains the p-value for the test. We have a highly significant result: $F(2,370) = 10.427, p < .000$. The populations do not each have the same mean.

To reach more detailed conclusions we might want to make a box plot. This can be done (as when making a histogram) with *Chart Builder*. The resulting output is displayed in Figure 6.11. The box plot suggests that third grade teachers score slightly lower. However, we should remind ourselves to not necessarily equate the 50th percentile with the mean. We can display the means for each sample along with some other descriptive statistics. Click on *Data* then *Split File*. We begin by clicking the bullet to the left of *Compare groups* before moving *Grade taught* into the window labeled *Groups Based on*. We then click *OK*. SPSS now reads our data set as

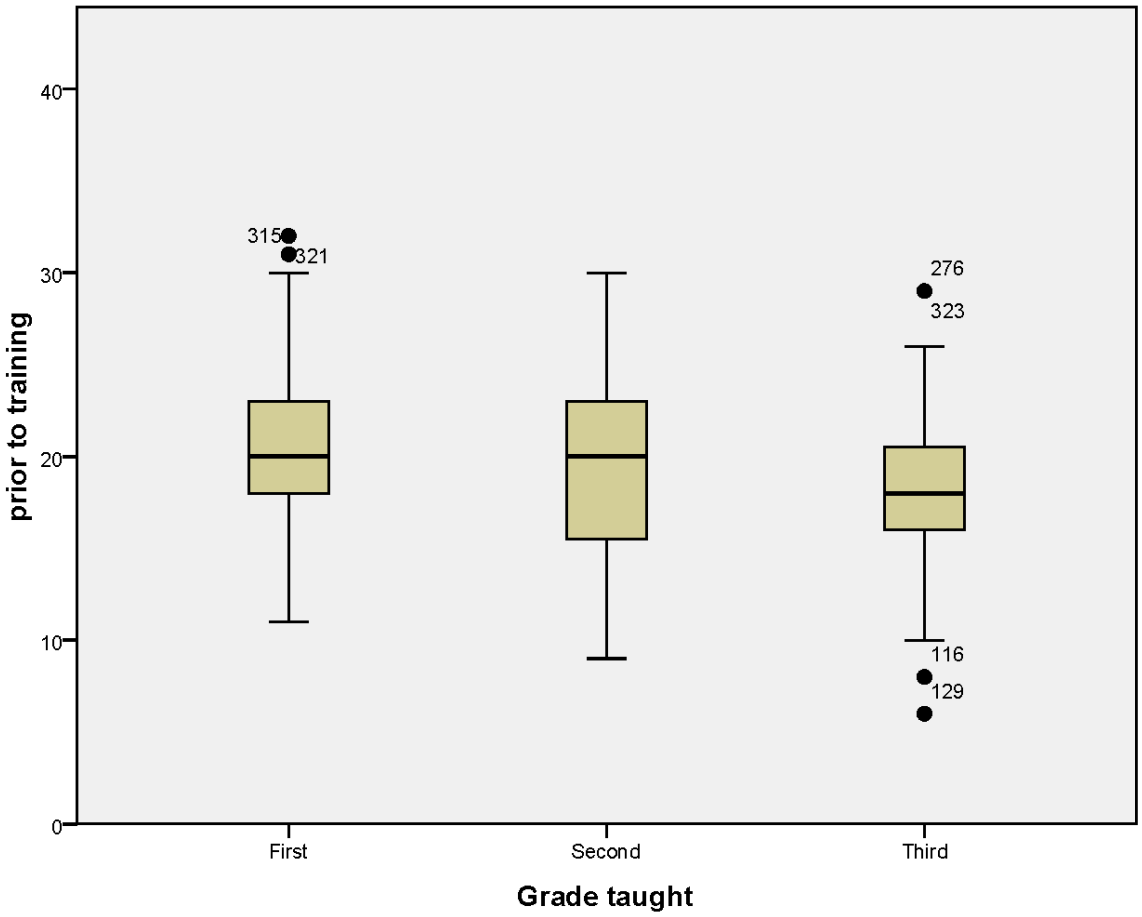


Figure 6.11. Box plots for pre test scores, grouped by 'grade taught'

three different data sets, one for each grade level. (Warning: make sure to undo the split groups command when you are done with this analysis.) To see some statistics for each group we click on *Analyze*, *Descriptive Statistics*, and then *Descriptives*. We then select our variable of interest, *prior to training*, and click *OK*. The results appear as shown in Figure 6.12. These results suggest that the lower the grade level taught, the higher the teacher's score on the pretest. We can check the significance of this observation by running pairwise t-tests, although we do not pursue this here. We are content with the significant ANOVA result, $F(2,370) = 10.427, p < .000$: the three population means are not all the same.

Here is another question that can be answered using ANOVA.

2. All teachers belong to one of five 'level of preparedness' groups. Does each group learn the same amount on average during the training session?

To answer this question we make use of the 'Improvement' variable that was created in example 3 of section 6.3. To test the ANOVA assumptions we plot an improvement histogram for each level of preparadness as shown in Figure 6.13. The assumptions are close enough to being met so we run ANOVA. The results are displayed in Figure 6.14. We get an insignificant result: $F(3,361) = .322, p = .810$. The well-prepared teachers don't necessarily get more or less out of the training session. We may even wish to interpret this as evidence for the quality of the training session.

Exercise 6.4. *Does each level of preparadness group score the same on average on the pre test?*

Descriptive Statistics

Grade taught		N	Minimum	Maximum	Mean	Std. Deviation
First	prior to training	150	11	32	20.71	4.306
	Valid N (listwise)	150				
Second	prior to training	111	9	30	19.37	5.034
	Valid N (listwise)	111				
Third	prior to training	112	6	29	18.17	4.139
	Valid N (listwise)	112				

Figure 6.12. SPSS table of descriptive statistics for 'prior to training' test scores, grouped by 'Grade level'

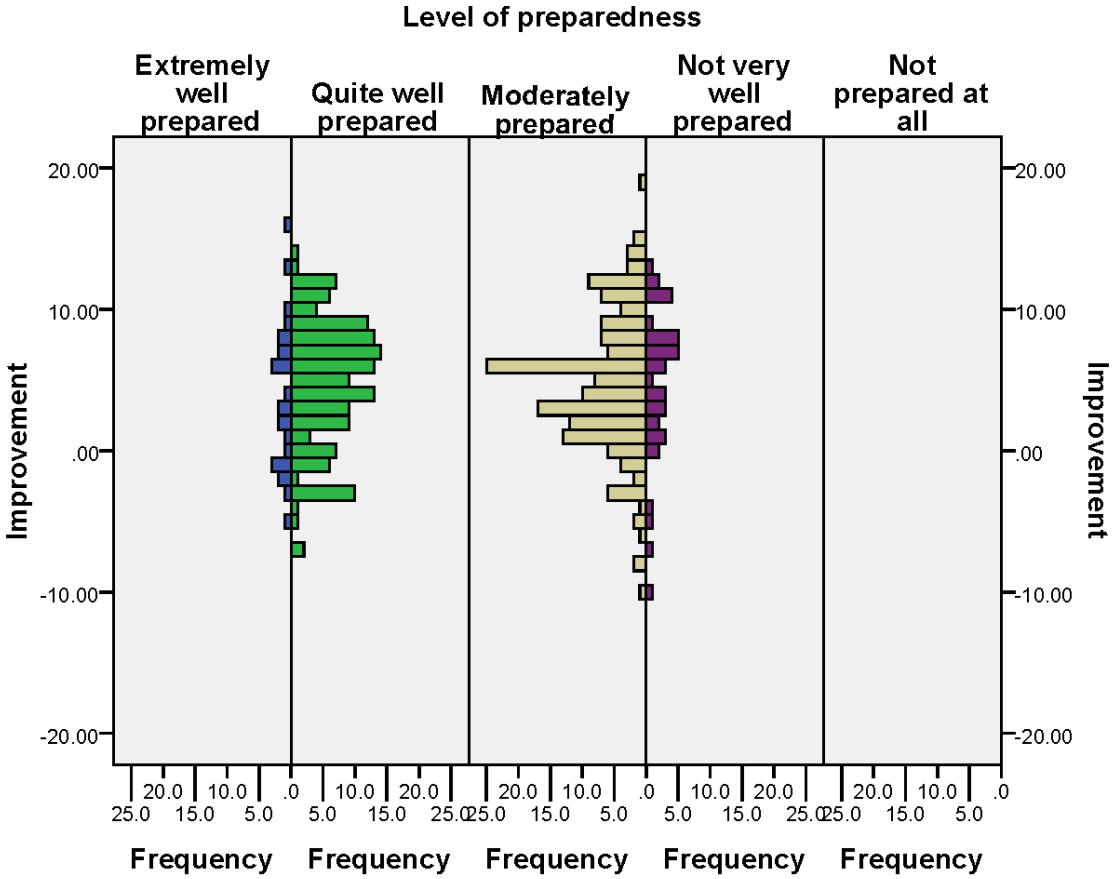


Figure 6.13. Histograms grouped by 'Level of preparedness', showing the sample distribution of 'Improvement', no teachers checked 'Not prepared at all'

ANOVA

Improvement

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	23.019	3	7.673	.322	.810
Within Groups	8611.764	361	23.855		
Total	8634.784	364			

Figure 6.14. ANOVA table for 'Improvement' grouped by 'Level of preparedness' as produced by SPSS

CHAPTER 7

REGRESSION WITH R

This chapter focuses on linear regression analysis and how standard procedures can be implemented within R. For an introduction to R see Chapter 3.

7.1 Basic review of linear regression

We have the following data

Y	x_1	x_2	$\dots$	x_p
Y_1	x_{11}	x_{12}	$\cdots$	x_{1p}
Y_2	x_{21}	x_{22}	$\cdots$	x_{2p}
Y_3	x_{31}	x_{32}	$\cdots$	x_{3p}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Y_n	x_{n1}	x_{n2}	$\cdots$	x_{np}

which consist of n corresponding observations for each of $p+1$ variables. We assume $n > p+1$ and also that the matrix

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ 1 & x_{31} & x_{32} & \cdots & x_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

has full rank. We also assume that the model

$$Y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \epsilon_i \tag{7.1}$$

is appropriate. The x observations are fixed and the observations Y_i depend linearly on the fixed x observations and a random error term ϵ_i. Typically the errors are assumed to be independent and identically distributed $\epsilon_i \sim N(0,\sigma^2)$. There are thus $p+2$ unspecified parameters. Focus on the following vector of $p+1$ parameters:

$$\beta = (\beta_0, \beta_1, ..., \beta_p).$$

It can be estimated with

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{7.2}$$

where X is as defined previously and $\mathbf{Y}$ is the vector of n observations of Y. We can use the vector of fitted values, $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p)$, to define the vector of residuals

$$\mathbf{r} = \mathbf{Y} - \mathbf{X}\hat{\beta},$$

which we can use to estimate σ^2. The following theorem nicely summarizes many related results.

Theorem 7.1. *If the errors are independent and normally distributed with mean zero and common variance σ^2 then*

1. *$\hat{\beta}_0, ..., \hat{\beta}_n$ and $\hat{\sigma}^2$ are MLEs and also jointly complete and sufficient.*
2. *$\hat{\beta}$ has a multivariate normal distribution with mean vector β and covariance matrix $\sigma^2(X'X)^{-1}$.*
3. *$n\hat{\sigma}^2/\sigma^2 =^d \chi^2(n-p-1)$.*
4. *$\hat{\beta}$ and $\hat{\sigma}^2$ are independent.*
5. *Each $\hat{\beta}_j$ is the UMVUE of β_j*
6. *$\tilde{\sigma}^2 = \mathbf{r}'\mathbf{r}/(n-p-1)$ is the UMVUE of σ^2.*

Proof. See [4], Theorem 15.4.4. □

Figure 7.1 and Figure 7.2 provide a graphical interpretation.

7.2 The fitted model

In the opening section of this chapter we introduced the (unfit) linear model

$$Y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \epsilon_i.$$

After estimating

$$\beta = (\beta_0, \beta_1, ..., \beta_p)$$

with

$$\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p) = \mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

(see (7.2)), we then have the fitted model

$$y(x_1, ..., x_p) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_p x_p.$$

It is simply a linear function of the x variables and it can be used for prediction. In the context of a fitted model the symbols $y, x_1, ..., x_p$ are variable names and do not represent observations of

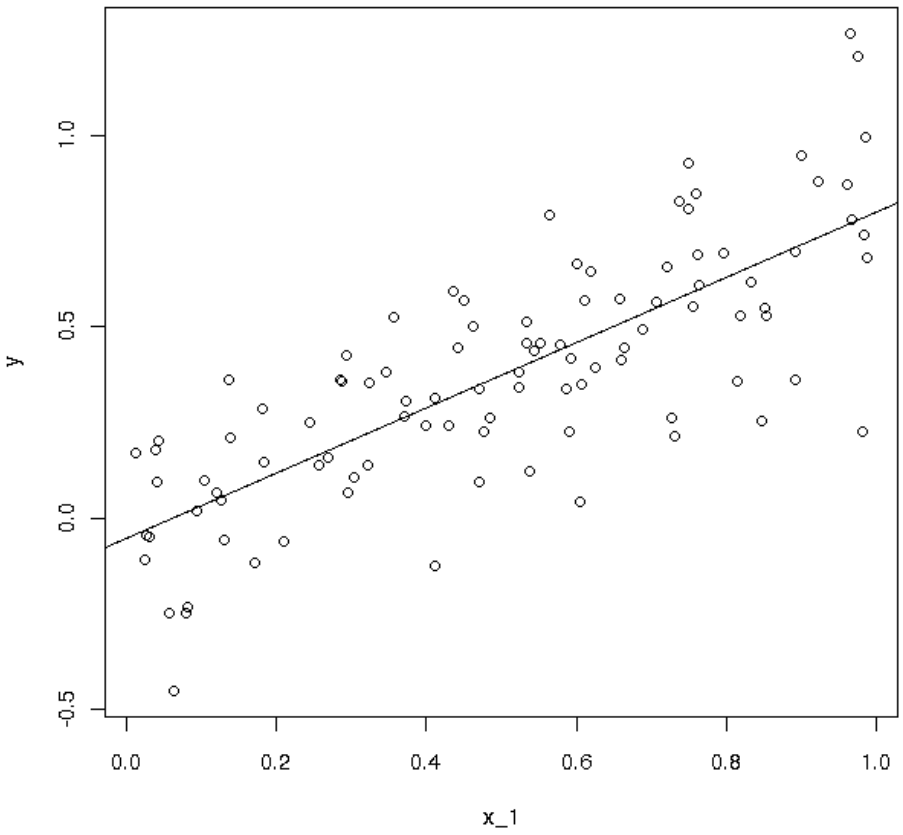


Figure 7.1. In case $p = 1$ we fit a line to the two-dimensional data

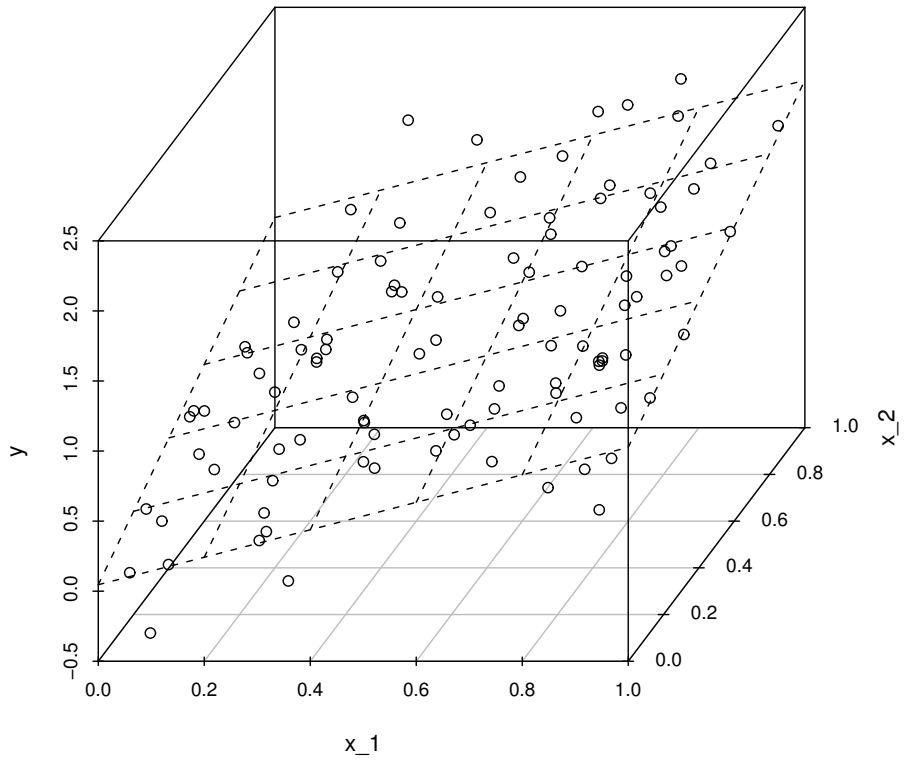


Figure 7.2. In case $p = 2$ we fit a plane to the three-dimensional data

data. Another important point is that the fitted coefficients are themselves random quantities. The fitted model is itself random. This section seeks to reinforce this fact and also introduce some basic R syntax for linear modeling.

For simplicity we temporarily restric to the case $p = 1$. That is we wish to model Y as a linear function of x. We assume the following.

- A linear relationship exists: $Y_i = \beta_0 + \beta_1 x_{i1} + \epsilon_i$.
- The errors are i.i.d. normal with common variance: $\epsilon_i = Y_i - (\beta_0 + \beta_1 x_{i1}) =^d N(0, \sigma^2)$.

In practice, these assumptions, although often reasonable, will not be exactly satisfied. While this is a possible source of error, it's important to distinguish this error from the error that is built into the model. To this end, we will begin with an artificial example; one where the regression assumptions are perfectly satisfied. We simulate some data.

```
> x_1=rnorm(19,10,3)
> y_1=1+.5*x_1+rnorm(19,0,1)
```

The fact that the observed x_1 values are normally distributed doesn't play much of a role here. Also, we arbitrarily set $\beta_0 = 1$, $\beta_1 = 0.5$ and $\sigma^2 = 1$. Keep in mind that in practice these quantities are unknown. Indeed, a goal of regression analysis is to estimate them.

In R, we can create a linear model and name it with the following syntax.

```
> LM_1=lm(y_1~x_1)
```

```
LM_1
```

now refers to a linear function $y(x_1) = \hat{\beta}_0 + \hat{\beta}_1 x_1$ that has been fit using the formula from (7.2). Think of it as a single observation of a random phenomenon. Since we are simulating, we can make more observations.

```
> x_2=rnorm(19,10,3)
> y_3=1+.5*x_3+rnorm(19,0,1)
> LM_2=lm(y_2~x_2)

> x_3=rnorm(19,10,3)
> y_3=1+.5*x_3+rnorm(19,0,1)
> LM_3=lm(y_3~x_3)
```

We now have three linear models

```
LM_1, LM_2, LM_3.
```

This is similar to having three observations of a random variable. Each fitted model (observation) arises from a new sample of bivariate data and the source of the randomness is always 'rnorm(19,0,1)'.

Of course in practice if we had three samples we would pool them and then fit a single linear model.

```
> x=c(x_1,x_2,x_3)
> y=c(y_1,y_2,y_3)
> LM=lm(y~x)
```

We have now defined four different linear models based on our simulated data. They can be plotted in a single graphic by first specifying the graphical parameters, and then for each model plotting the data and adding its regression line. Titles and colors are specified as well and the output is displayed in Figure 7.3.

```
> par(mfrow=c(2,2))  ### setting graphical parameters


> plot(x_1,y_1,col='red',main='First Sample')
> abline(LM_1,col='red')

> plot(x_2,y_2,col='green',main='Second Sample')
> abline(LM_2,col='green')

> plot(x_3,y_3,col='blue',main='Third Sample')
> abline(LM_3,col='blue')


> plot(x,y,main='All Three Samples Pooled')
> abline(LM)
```

Perhaps an even more striking visual is one with all the relevant information superimposed on a single plot. This is obtained via the following:

```
> par(mfrow=c(1,1))

> plot(x,y,main='Randomness of the Fitted Models')   ### a quick way
to set the correct window size for the graph
> points(x_1,y_1,col='red')
> points(x_2,y_2,col='green')
> points(x_3,y_3,col='blue')

> abline(LM)
> abline(LM_1,col='red')
> abline(LM_2,col='green')
> abline(LM_3,col='blue')

> abline(1,.5,lwd=2)       ###'lwd' stands for 'line width'
```

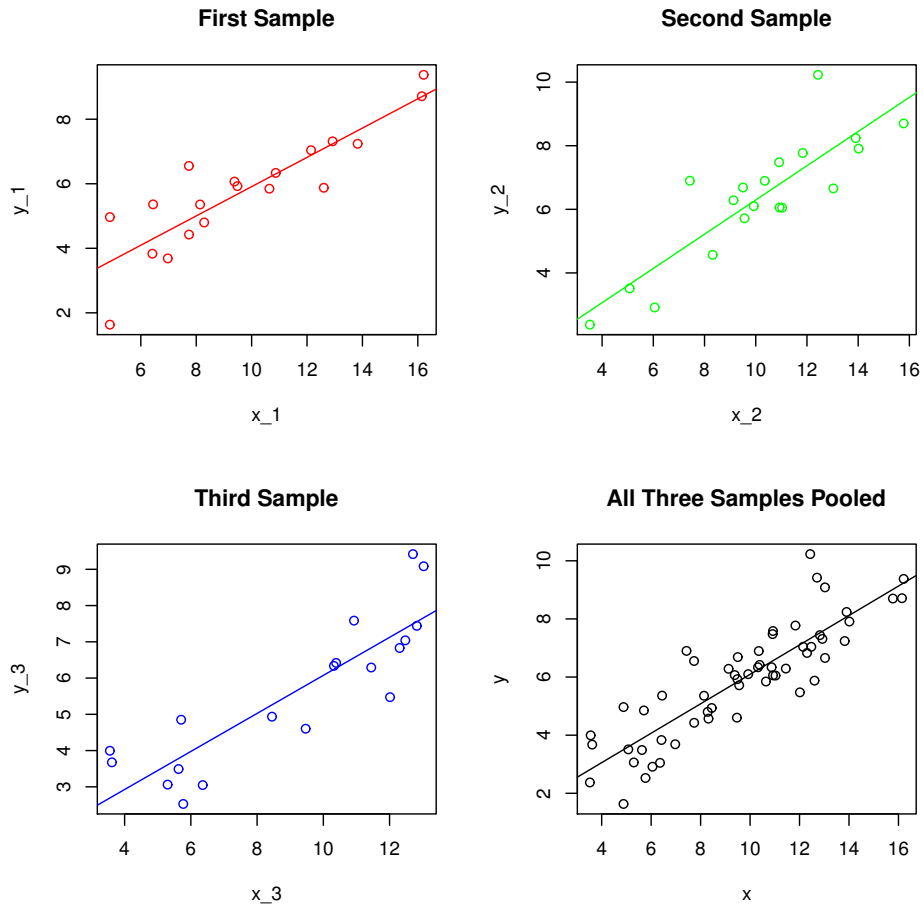


Figure 7.3. Each fitted model is different

The output is displayed in Figure 7.4. Again, in practice all we would have is the pooled data and its associated fitted model LM. The graphic of just the pooled data and its associated regression line is obtained with the following commands and the output is displayed in Figure 7.5.

```
> plot(x,y,main='All that the Experimenter Sees')
> abline(LM)
```

Without knowing the true relationship between $E(Y_x)$ and x how can we comment on the accuracy of our fitted model? Many useful statistics have been developed, some of which are displayed by using R's 'summary' command, which can act on a named linear model.

```
> summary(LM)

Call:
lm(formula = y ~ x)
```

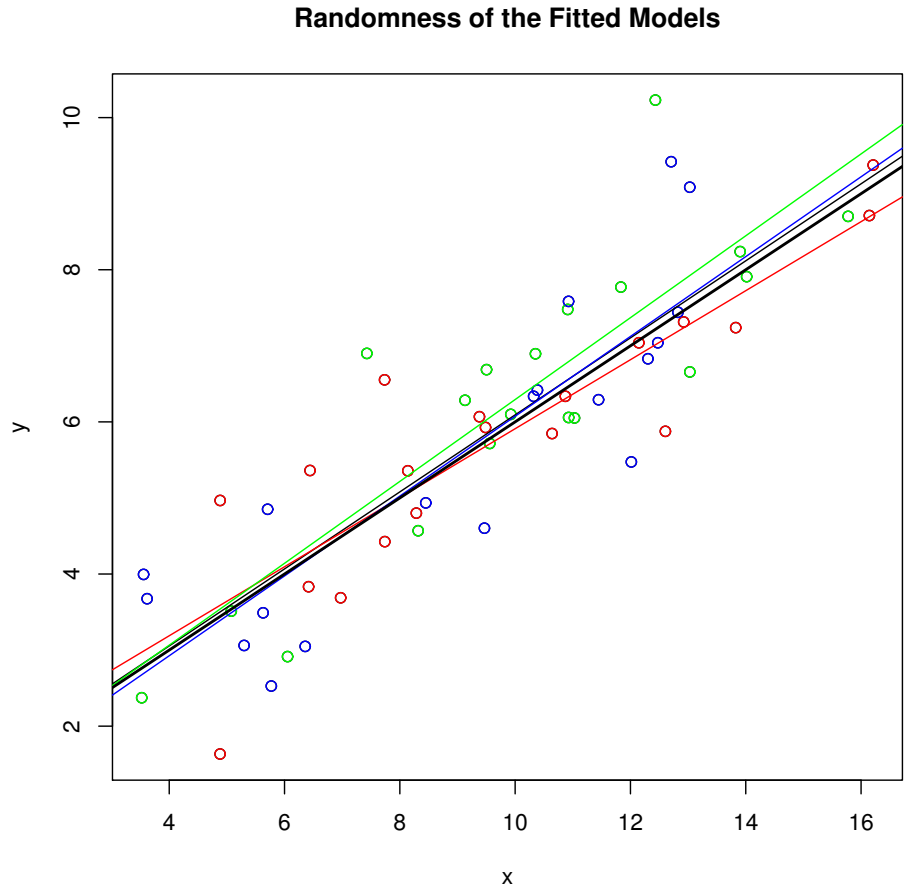


Figure 7.4. The samples and associated linear models are color coded. The linear model for the pooled data is the thin black line. The true linear relationship between $E(Y_x)$ and x is represented by the thick black line.

```
Residuals:
    Min      1Q  Median      3Q     Max
-1.8675 -0.5331 -0.1964  0.6322  2.9071

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.02945    0.40637   2.533   0.0142 *
x            0.50627    0.03985  12.706   <2e-16 ***
---
Residual standard error: 0.9819 on 55 degrees of freedom
Multiple R-squared: 0.7459,     Adjusted R-squared: 0.7413
F-statistic: 161.4 on 1 and 55 DF,  p-value: < 2.2e-16
```

In the sections that follow we will gain familiarity with the above printout.

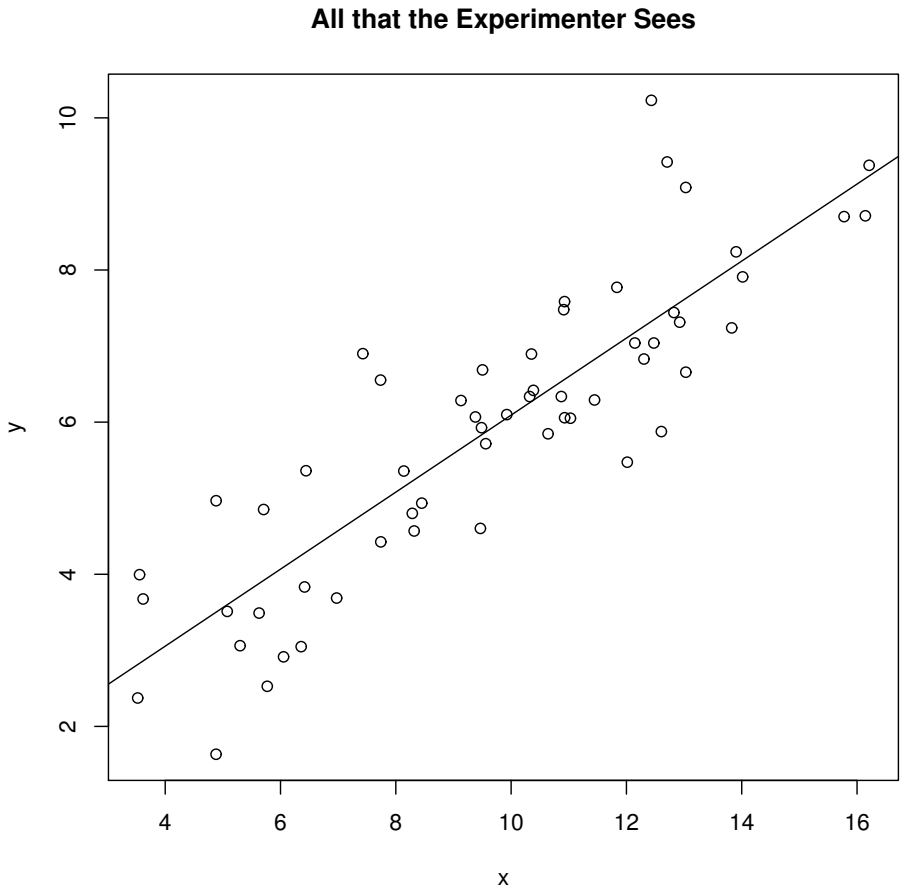


Figure 7.5. The true linear relationship between $E(Y_x)$ and x is unobservable

Exercise 7.1. *Plot the following two dimensional data:*

$x = c(2,3,4,5,6,7,8,9,10,11,12,13,14,15)$*, and*

$y = c(.7,1.2,1.4,1.5,1.6,1.9,2.0,2.3,2.1,2.2,2.5,2.6,2.6,2.6)$.

Do the standard regression assumptions appear to hold?

7.3 Checking the regression assumptions

In the previous section we learned how to use R to fit a linear model and also to print out a list of summary statistics. These statistics are only as reliable as the assumptions that they are based on,

- linear relationship: $Y_i = \beta_0 + \beta_1 x_{i1} + \epsilon_i$
- errors i.i.d. normal: $\epsilon_i = Y_i - (\beta_0 + \beta_1 x_{i1}) =^d N(0, \sigma^2)$,

thus we had better develop some procedures for checking these assumptions. We do not expect for these assumptions to be satisfied exactly, thus we will conduct a search for evidence that strongly refutes the assumptions. The stronger we search, and the less contradictory evidence we find, the more confidence we will then have in the statistics from the summary.

We illustrate some of the procedures by working with real data. There are many readily available datasets stored within R. Type swiss to access one of these.

```
> swiss
```

R describes this dataset as, 'Standardized fertility measure and socio–economic indicators for each of 47 French–speaking provinces of Switzerland at about 1888'. Additional details can be obtained by typing

```
>?swiss
```

So we have at our disposal observations of six different variables for each of forty seven different counties. For future reference we can name these variables.

```
> fert=swiss[,1]      ###common standardized fertility measure
> agric=swiss[,2]     ###percent of males working in agriculture
> exam=swiss[,3]      ###percent of draftees receiving highest mark on
                          entrance exam
> educ=swiss[,4]      ###percent of draftees educated beyond primary school
> cath=swiss[,5]      ###percent Catholic (as opposed to Protestant)
> IM=swiss[,6]        ###measure of live births who live less than one year
```

Lets examine the relationship between 'agric' and 'fert' by creating a linear model

```
>LMag=lm(fert~agric)
```

The graphics can be displayed by typing

```
>plot(agric,fert)
>abline(LMag)
```

which results in the output displayed in Figure 7.6 For our above 'LMag' linear model, based on the plot, there is no obvious evidence for rejecting the existence of a linear relationship between $E(Y_x)$ and x.

We use the residuals to gain insight into the errors and to assess the independent-errors assumption. A common technique involves plotting the residuals as a function of the explanatory observations. Under the independent–errors assumption we can expect this plot to be devoid of patterns. On the other hand, a pattern suggests that the errors have covariance, and are thus not independent.

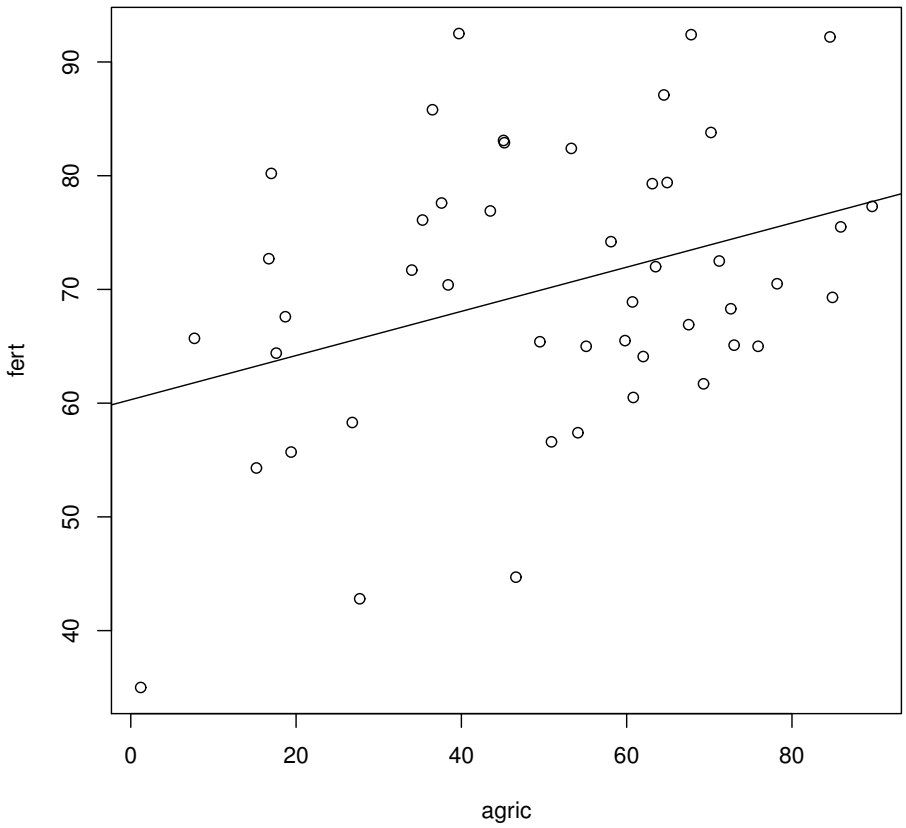


Figure 7.6. Bivariate data and least squares regression line

Here is how R can make such a residuals plot. First we label the residuals vector for future reference. Note how this is done through the use of the double dollar sign.

```
> res=LMag$$residuals
> plot(agric,res)
```

The output is displayed in Figure 7.7 There is no obvious pattern and thus we have no solid reason for rejecting the independent-errors assumption, yet.

If the errors display covariance with another variable from the dataset, that is reason enough to doubt the independent–errors assumption. So we might want to plot the residuals vector against other variables as well. Another option is to plot the residuals vector against the fitted values, $\hat{\beta}_0 + \hat{\beta}_1 x_{i1}$. One should not plot the residuals against the Y_i values though (see Exercise 7.2). It remains to check for normality of the errors. Once again, we turn to the residuals vector for information. We can make a histogram using

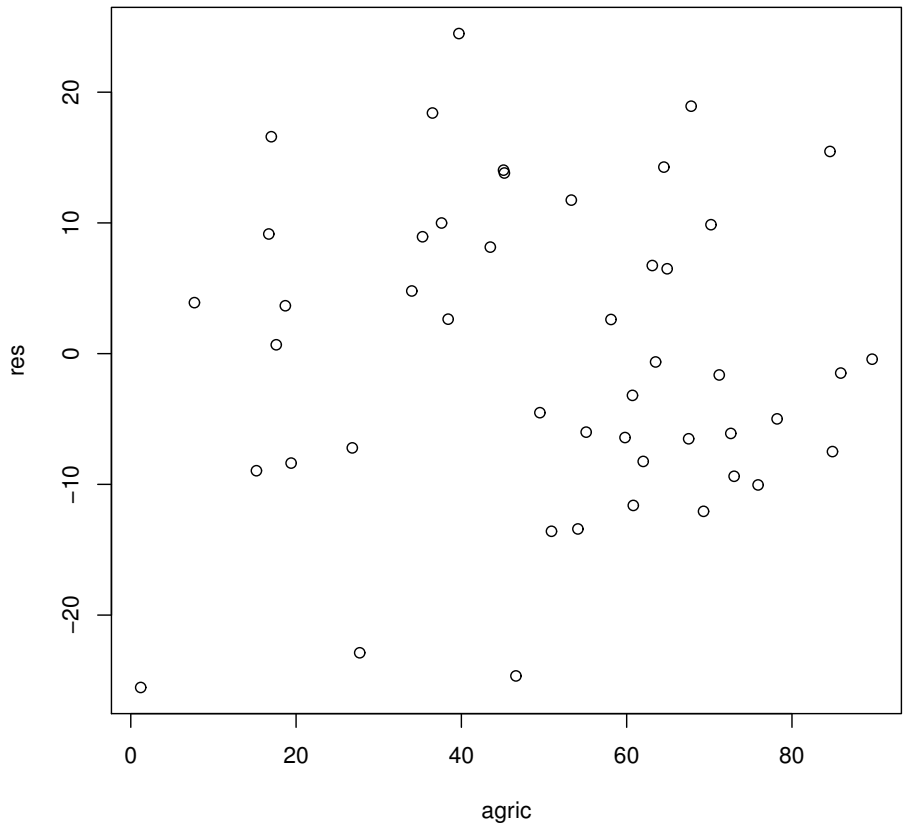


Figure 7.7. Residuals as a function of the sole predictor 'agric'

```
> hist(res)
```

and the output is shown in Figure 7.8. The histogram is not wildly non-normal. A Q-Q plot can give some information as well as seen in Figure 7.9.

```
> qqnorm(res)   ### creates a quantile quantile plot
                    against a normal random variable
```

The line is absent of obvious curves which would indicate non-normality. Just to be certain, we run the Shapiro–Wilk normality test on the residual data.

```
> shapiro.test(res)

        Shapiro-Wilk normality test

data:  res
W = 0.9805, p-value = 0.6121
```

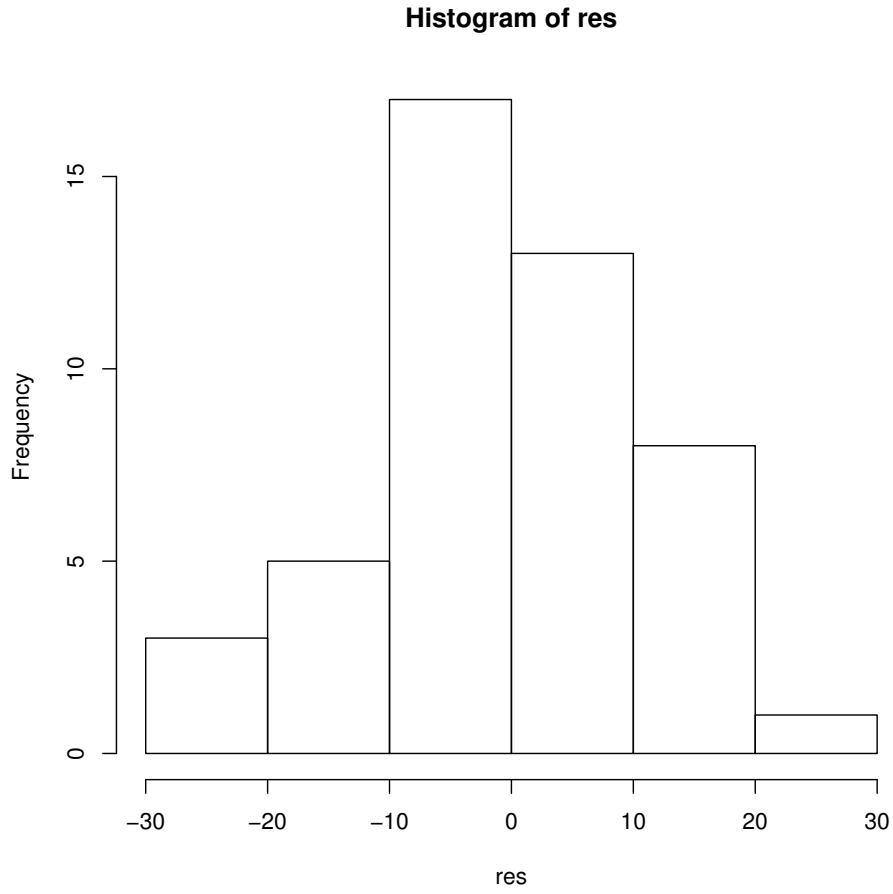


Figure 7.8. Histogram for the Residuals

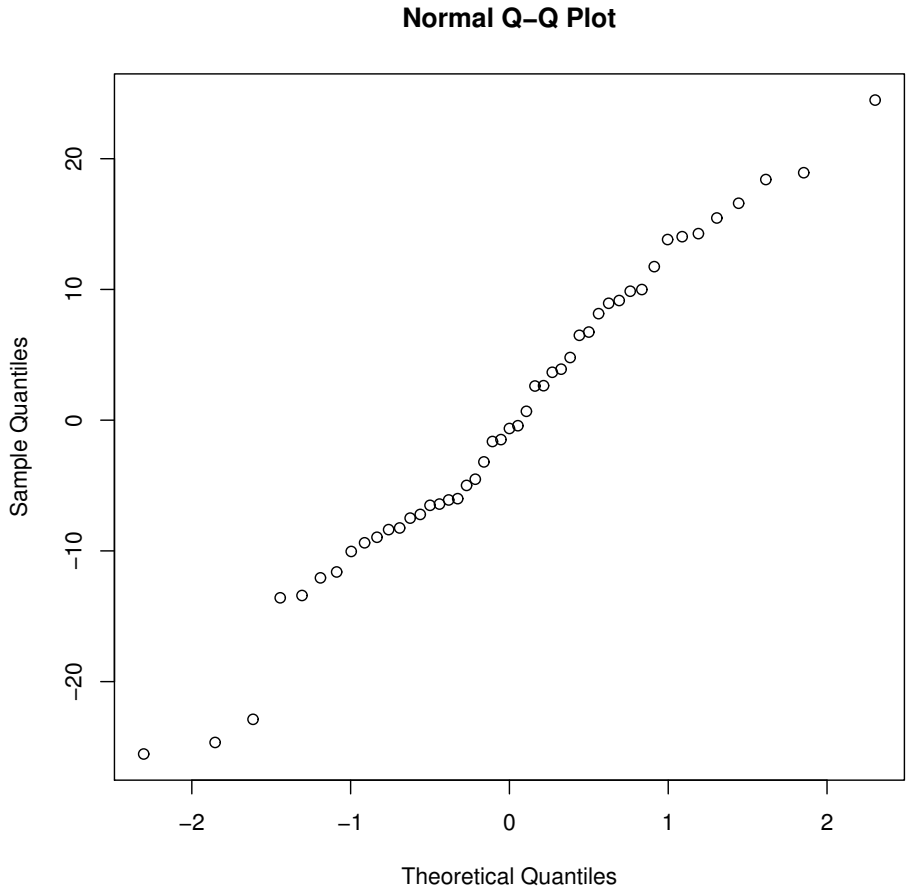


Figure 7.9. Q-Q plot with normal quantiles on the x-axis and the residuals on the y-axis

The null-hypothesis of this test is normality, and thus our insignificant p-value, coupled with the fact that we don't expect the residuals to perfectly represent the normality of the errors anyway, leaves us with no obvious reason to reject the normality assumption.

In fact, neither this test, nor the histogram, Q-Q plot, residuals plot, or bivariate data plot provided clear evidence to reject any of our assumptions. We thus turn to the summary statistics with confidence.

We will give them each individual treatment in the sections to come.

Exercise 7.2. *For 'LMag' as defined above plot the residuals 'res' as a function of the observed fertility rate Y_i. Explain why any pattern that results is not reason for doubting the independent errors assumption.*

Exercise 7.3. *Pick two variables from the swiss dataset and run a simple regression. Before summarizing the model, check the assumptions using as many techniques as you can. Do you feel confident reporting the statistics fromt the summary?*

7.4 Interpreting the t statistic

In this section we will show how the t statistic arises in the context of regression. Actually, for a given linear model, there will be multiple t statistics; one for each of the beta parameters.

We begin with another look at the swiss data-set variables fert and agric. The bivariate plot with regression line is shown again in Figure 7.10. In the last section we found no reason to reject the standard regression assumptions. Thus, we feel some confidence in the validity of the statistics as displayed in the model summary.

```
> summary(LMag)
```

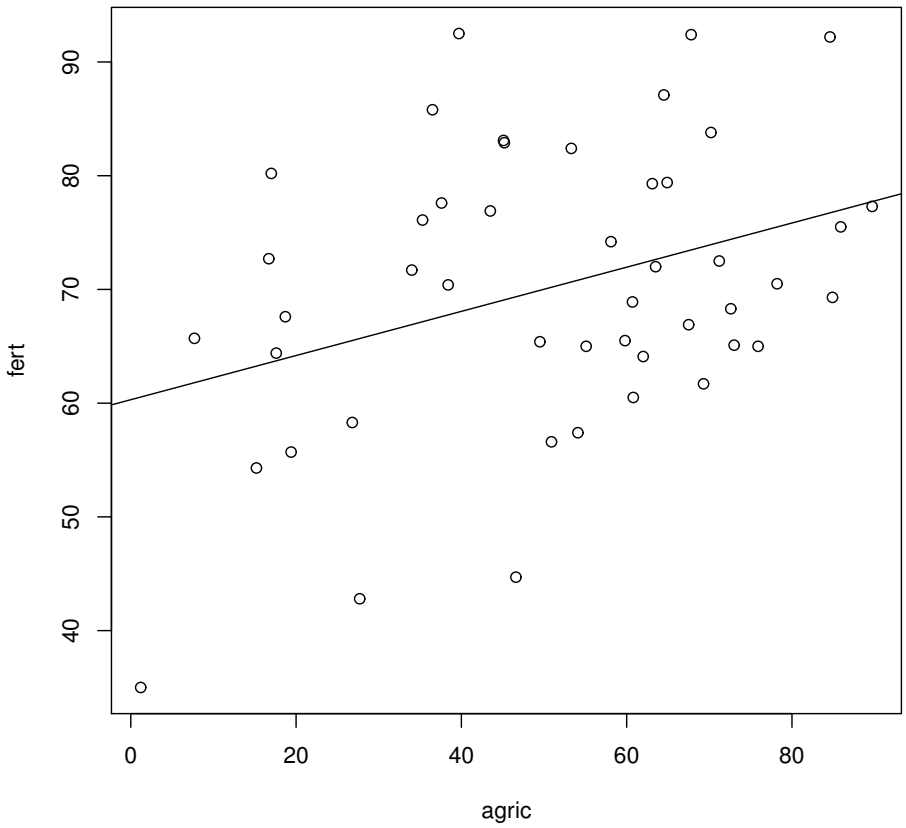


Figure 7.10. Bivariate data and least squares regression line

```
Call:
lm(formula = fert ~ agric)

Residuals:
     Min       1Q   Median       3Q      Max
-25.5374  -7.8685  -0.6362   9.0464  24.4858

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 60.30438    4.25126  14.185   <2e-16 ***
agric        0.19420    0.07671   2.532   0.0149 *
---
Residual standard error: 11.82 on 45 degrees of freedom
Multiple R-squared: 0.1247,     Adjusted R-squared: 0.1052
F-statistic: 6.409 on 1 and 45 DF,  p-value: 0.01492
```

Notice that each coefficient has an associated t value. We now show how this t-value is computed.

We know from Theorem 7.1 that $\hat{\beta}$ has a multivariate normal distribution with mean vector β and covariance matrix $\sigma^2(X'X)^{-1}$. A consequence (see Johnson and Wichern, Result 4.4) is that the the marginals, the $\hat{\beta}_j$'s, are distributed as $N(\beta_j, \sigma^2(X'X)^{-1}_{jj})$. Standardizing results in

$$\frac{\hat{\beta}_j - \beta_j}{\sigma^2(X'X)^{-1}_{jj}} =^d N(0,1).$$

Substituting $\tilde{\sigma}^2$ in for σ^2 then by way of Theorem 7.1 leads to the following conclusion:

$$\frac{\hat{\beta}_j - \beta_j}{\tilde{\sigma}^2(X'X)^{-1}_{jj}} =^d t(m-n-1).$$

We state this as a theorem.

Theorem 7.2. *If $Y_1, Y_2, ..., Y_m$ are independent with $Y_i \sim N\left(\sum_{j=0}^{n} \beta_j x_{ij}, \sigma^2\right)$, then*

$$\frac{\hat{\beta}_j - \beta_j}{\tilde{\sigma}^2(X'X)^{-1}_{jj}} =^d t(m-n-1).$$

This holds regardless of the amount of explanatory variables, and allows us to test whether a given fitted coefficient, $\hat{\beta}_j$, is significantly different from zero. The test performed is two-sided and it is summarized across the appropriate line under 'coefficients' in the summary.

For rapid viewing, significant results are denoted with asterics. A single asterix signals significance at the .05 level, a double asterix at the .01 level and a triple asterix at the .001 level. A single dot signals significance only at the .1 level.

As we shall see in the following sections, when there are many explanatory variables, the t-values for each coefficient provide reason for keeping or omiting that variable from the model.

In our case we conclude (since we have accepted the regression assumptions) that the population intercept β_0 is highly significantly different from zero, and that the population slope β_1 is significantly different from zero.

Even had we done a one-sided test for the slope, rejecting the null only for aberantly high (under the null) values of $\hat{\beta}_1$, our t value was still high enough to reject the null. Thus, we even have solid evidence that the population slope β_1 is positive.

Caution is advised, however. These results are meant to be interpreted within the context of the particular parametric model that has been used, namely $fert = \beta_0 + \beta_1 agric + N(0, \sigma^2)$. It is incorrect to conclude that in the real world, specifically French speaking Switzerland, that the more agriculatural provinces can be expected to be more fertile. Such a broad interpretation mistakenly assumes that the model is well fit. A quick look back at the plot is enough to convince the reader that the relationship between $fert$ and $agric$ is dominated by the error term. Perhaps $agric$ is not the only measurable variable that exerts a linear influence on $fert$.

Indeed, in the next section we will see that after including $educ$, $cath$, and IM, in addition to $agric$ as predictor variables in the linear model, that all of the coefficients will be deemed significantly different from zero, and remarkably the sign of the estimated coefficient for $agric$ will have become negative.

Exercise 7.4. *Pick two variables from the swiss dataset and run a simple regression analysis. If the assumptions are satisfied then compute the t statistic for the slope. Do so the long way using the formula in Theorem 7.2. Remember that R can be used as a calculator. Compare your answer with the automated result from linear model summary.*

7.5 Interpreting R^2

In the previous section we mentioned that the model $fert = \beta_0 + \beta_1 agric + N(0, \sigma^2)$ is not well fit to the data. In this section we make this notion more precise. We introduce the R^2 and adjusted $-R^2$ statistics.

As an introduction take a look at the graphics for three separate linear models as shown in Figure 7.11. Which model is the most useful? There is still a lot of unexplained variance in the top model, less unexplained variance in the middle model, and almost all the variance is explained in the bottom model. This does not mean that the bottom model is the most useful though. Might we not explain Y_3 just as well by saying it's a random variable with mean zero? In other words,

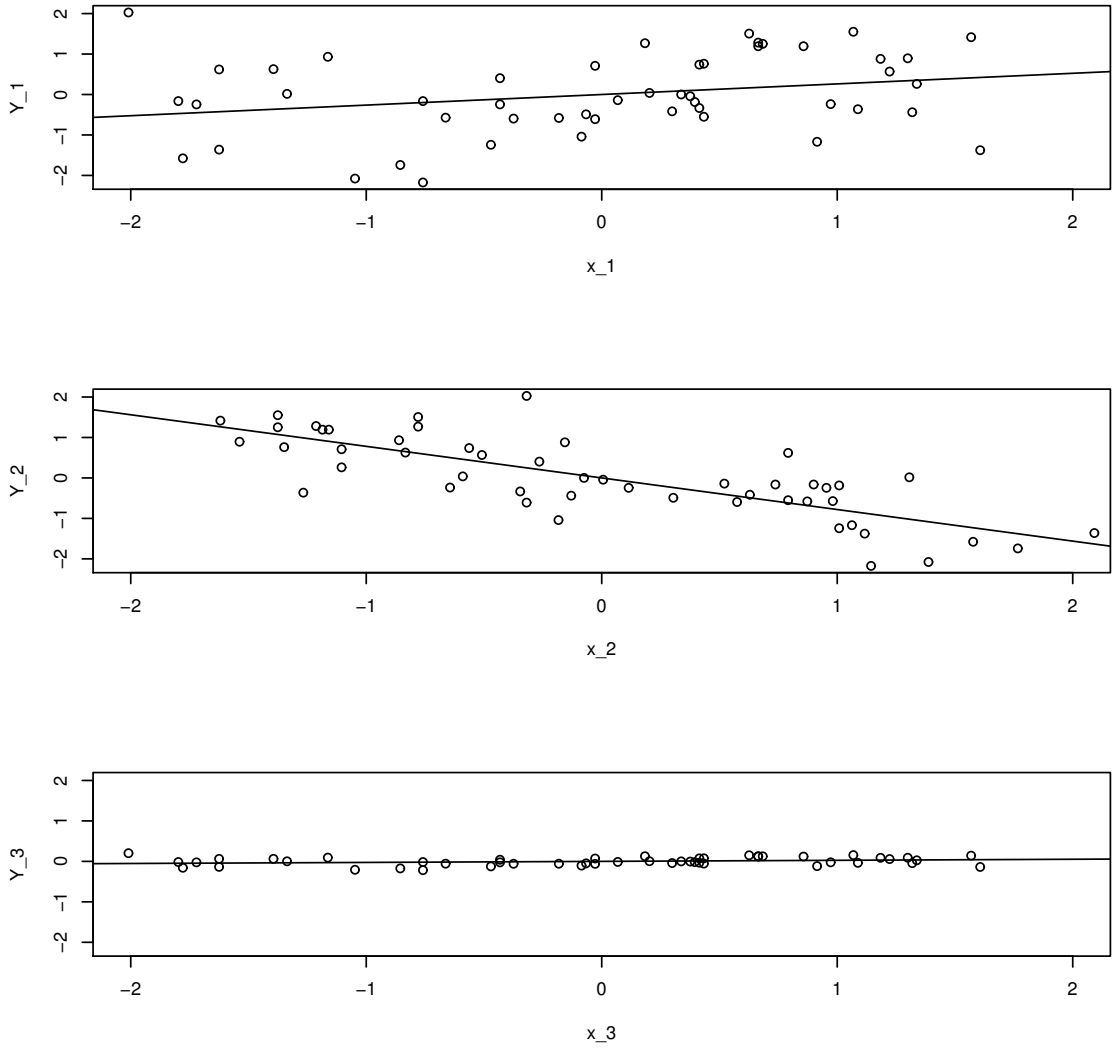


Figure 7.11. Why is model 2 the best?

x_3 does not tell us much about Y_3. On the other hand, x_2 plays a significant role in the middle model. Imagine a future observation of x_2 where Y_2 isn't known. To best predict Y_2 should the second model be used? Keep these considerations in mind as we define the following terms. Our aim is to quantify the goodness of fit of a model.

With σ_ϵ denoting the standard deviation of the error term and σ_Y the standard deviation of Y we define the following.

Definition 7.3. *Theoretical* R^2

$$1 - \sigma_\epsilon / \sigma_Y$$

This will be our gauge for the goodness of fit of a linear model. Well–fit models have theoretical R^2 values close to one. Poorly fit models have theoretical R^2 values close to zero. Of course this theoretical quantity must be estimated from the data. The R^2 statistic is a simple estimator. We define it in terms of the fitted values $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1}$ and the sample mean for Y namely $\bar{Y}$.

Definition 7.4. R^2

$$R^2 := 1 - \frac{\sum_{i=1}^{n} (Y_i - \hat{Y})^2}{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}$$

.

R^2 performs well when there are only a few explanatory variables in the model. However, with too many explanatory variables the R^2 statistic becomes artificially high. Consider the following simulation, where each random sample of size ten is completely independent from the others.

```
> y=rnorm(10)
> x_1=rnorm(10)
> x_2=rnorm(10)
> x_3=rnorm(10)
> x_4=rnorm(10)
> x_5=rnorm(10)
> x_6=rnorm(10)
> x_7=rnorm(10)
> x_8=rnorm(10)
> x_9=rnorm(10)
```

We will fit linear models using various subsets of explanatory variables and observe how the R^2 statistic fluctuates. The syntax for adding additional explanatory variables to a model is simple and displayed in the following table. We write and use the following function

```
>r2.val=function(x) summary(x)$r.squared
### the dollar sign $ restricts attention to what follows it
```

in order to quickly abstract R^2 values from specified, fitted models. Note that R^2 is labeled in R as 'Multiple R–squared'. The following table summarizes our results.

Model	R^2
lm(y ~ x_1)	2.735264e-06
lm(y ~ x_1+x_2)	0.1369587
lm(y ~ x_1+x_2+x_3)	0.1482091
lm(y ~ x_1+x_2+x_3+x_4)	0.3160829
lm(y ~ x_1+x_2+x_3+x_4+x_5)	0.5254409
lm(y ~ x_1+x_2+x_3+x_4+x_5+x_6)	0.5406889
lm(y ~ x_1+x_2+x_3+x_4+x_5+x_6+x_7)	0.9770266
lm(y ~ x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8)	0.997908
lm(y ~ x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9)	1

Despite the independence of our samples we have high R^2 values indicating well fit models when the number of explanatory variables approaches the sample size. This signals the fact that R^2 is a biased estimator for the theoretical R^2. Adjusting for this bias results in the adjusted R^2 statistic, which is unbiased for the theoretical R^2.

Definition 7.5. *Adjusted R^2*

$$\text{adjusted } R^2 = 1 - \frac{\left(\sum_{i=1}^{n}(Y_i - \hat{Y})^2\right)/(n-p-1)}{\left(\sum_{i=1}^{n}(Y_i - \bar{Y})^2\right)/(n-1)}.$$

We can add a column of adjusted R^2 values to our previous table, to see how it guards against overfitting.

Model	R^2	adjusted R^2
lm(y ~ x_1)	2.735264e-06	-1.249969e-01
lm(y ~ x_1+x_2)	0.1369587	-0.1096245
lm(y ~ x_1+x_2+x_3)	0.1482091	-0.2776863
lm(y ~ x_1+x_2+x_3+x_4)	0.3160829	-0.2310508
lm(y ~ x_1+x_2+x_3+x_4+x_5)	0.5254409	-0.06775794
lm(y ~ x_1+x_2+x_3+x_4+x_5+x_6)	0.5406889	-0.3779333
lm(y ~ x_1+x_2+x_3+x_4+x_5+x_6+x_7)	0.9770266	0.8966199
lm(y ~ x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8)	0.997908	0.9811713
lm(y ~ x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9)	1	NaN

The first thing to notice is that for poorly fit data the adjusted R^2 can even be negative. Secondly, notice the positive improvement: adjusted R^2 hovers nearer to zero as it should. Except when the model becomes extremely overfit, then we see even the adjusted R^2 approaching 1 when it shouldn't. In the section of this chapter on bootstrapping, Section **??**, we construct a confidence

interval for the variance of the R^2 statistic. The procedure can be used for the adjusted R^2 statistic as well.

Exercise 7.5. *For the swiss dataset try to model - using the 'lm' command - fertility as a linear function of the remaining indicators (the other variables). You can include as many or as few of the predictors as you wish, but make sure to justify how you arrived at your final choice of model.*

Exercise 7.6. *For the same dataset try modeling one of the predictors this time, as a function of a set of the other variables. For example, you might want to try to predict infant mortality as a function of the other variables.*

7.6 Interpreting the F statistic

This section demonstrates how the F statistic is useful in the context of linear regression. The first subsection demonstrates how the F statistic arises. The middle subsections focus on useful R-skills such as importing data and saving programs. The last subsection shows how the F statistic can be used to compare nested models.

7.6.1 How the F statistic arises

After completing the exercises from the previous section we now have some familiarity with the dilemna of choosing between various subsets of explanatory variables to be used in the linear model. Various procedures have been developed that choose a subset to be used, however none of this procedures can be shown to be optimal in all situations. Oftentimes, one is faced with choosing between two nested models, $Y = \beta_0 + \beta_1 x_1 + ... + \beta_{n_1} x_{n_1}$ and $Y = \beta_0 + \beta_1 x_1 + ... + \beta_{n_1} x_{n_1} + ... + \beta_{n_2} x_{n_2}$. By nested we mean that the first, smaller model is contained within the second, larger model, which also employs the additional predictors $x_{n_1+1}, x_{n_1+2}..., x_{n_2}$ as explanatory variables. We want a systematic way of deciding whether the larger model is significantly better than the smaller model.

We will present here a test with the null hypothesis stating that the additional coefficients $(\beta_{n_1+1}, ..., \beta_{n_2})$ are all zero.

Under this assumption we use $SSEn = \sum_{i=1}^{n} (\hat{Y}_i - \bar{\mathbf{Y}})^2$ to define the following statistic:

$$SSEn_1 - SSEn_2.$$

The postscript n_1 signals that the fitted values come from the smaller model. The postscript n_2 signals that the fitted values come from the larger model.

Under the alternative we expect the statistic to grow and we thus plan to reject the null when the statistic is large. In fact, we can manipulate the statistic so that under the standard assumptions it is distributed as an F.

Theorem 7.6. *If the design matrix has full rank and the standard regression assumptions are met then*

$$\frac{(SSEn_1 - SSEn_2)/(n_2 - n_1)}{SSEn_2/(m - n_2 - 1)} =^d F_{n_2-n_1,m-n-1} \tag{7.3}$$

Furthermore, the test that rejects $H_0 : (\beta_{n_1+1}, ..., \beta_{n_2}) = (0, ..., 0)$ *when the statistic is large is equivalent to the generalized likelihood test.*

Proof. See Johnson and Wichern [12] Result 7.6. □

The α level for this test can be determined based on considerations of the cost of adding the extra predictors, and the plan is to adopt the larger model upon rejection of the null.

The special case of $n_1 = 0$ is worth noting. This compares no linear model (just the mean of the observed y's) against a larger linear model. The F statistic then informs us whether or not the large linear model is significantly better than no model at all. If it is significantly large (p-value less than α) then we conclude that the model is indeed useful (significantly better than no model). We can think of the p-value as reporting the probability, given that the predicters in question are independent of y, of obtaining data as extreme or more extreme than that observed. Thus the p-value is an inverse measure of the extremity of the data, given that Y is really not a function of any of the x's. The lower the p-value the stronger the argument for including at least one of the considered predicters in the final model. It is this special case where $n_1 = 1$ that R automatically includes in the summary of a given linear model.

Soon we will illustrate how the F-statistic can be used as a guide for model selection, but first we learn some new R skills.

7.6.2 Importing data into R

We will be working with a dataset titled 'Late80sCdata.csv' and we will assume that it has been saved to a directory. R should already be pointed toward a given directory. To find out which one simply type

```
>getwd()  ### stands for 'get working directory'
[1] "C:/Documents and Settings/u0274545/My Documents"
```

If you're working on a PC the output may be similar to the above. It shows that the current working directory is the 'My Documents' folder. To change directories, to say the Desktop, we can type the following.

```
> setwd("C:/Documents and Settings/u0274545/Desktop")
### 'setwd' stands for 'set working directory'
```

The argument for the function 'setwd' is the path for the directory you wish to use.

Now, we wish to access the Late80sCdata set which is saved in 'comma separated variable' format (.csv) on the desktop. This can be done as follows. Note how we name the data set within R as simply Cdata.

```
> Cdata=read.csv("Late80sCdata.csv",
colClasses = c("character","character",rep("numeric",47)))
```

The first argument is the name of the file. The second argumet (colClasses) specifies the types of variables in each column. In our case the first two variables are character variables and the remaining are numeric. We communicate this fact to R up front in order to avoid problems later on.

This dataset comes from a study titled *Geographic study of mortality, biochemistry, diet and lifestyle in rural China*. For more information see Oxford University's webpage for the study at `http://www.ctsu.ox.ac.uk/~china/monograph/`.

The original data involves observations from 69 different rural counties within China. For each county 639 variables were measured. The dataset we have loaded into R is a simplification. Data from only 64 of the counties, choosen because they had complete data are presented. Also, only 49 of the variables are presented, chosen because they are interesting and easy to understand. The names of the variables can be shown within R.

```
> names(Cdata)
 [1] "County"
 [2] "Region..N.for.North..S.for.South."
 [3] "Latitude"
 [4] "Income..per.capita"
 [5] "Industrial.Production..measure.of...Per.Capita"
 [6] "Distance.to.Nearest.City..km."
 [7] "Literacy..percentage..males."
 [8] "TV..percent.who.watch.several.times.per.week"
 [9] "Food.Shortages..severe..duration..months...over.last.30.years"
[10] "Alcohol.Consumption..percentage.who.have.ever.been.regular.drinkers"
[11] "Aflatoxin.M1.in.urine..pg.mg.creatinine."
[12] "Tobacco.Smoke..a.biomarker..Cotinine..was.measured.in.females.only
..percent.of.females.measuring...20.ng.mL.reported"
[13] "Pickled.Vegetables.frequency..average.number.of.days.per.year.eating
```

```
.salt.preserved.vegetables"
[14] "Green.Vegetables.frequency..average.days.per.year.eating.green.vegetables"
[15] "Legumes.frequency..average.days.per.year.eating.legumes..mostly.soy."
[16] "Smoked.Food..percent.of.people.having.ever.eaten.smoked.foods"
[17] "EPA..fatty.acid..Intake..mg.day.reference.man."
[18] "DHA..fatty.acid..Intake..mg.day.reference.man."
[19] "Dietary.Cholesterol.intake..mg.day.reference.man."
[20] "Saturated.Fatty.Acid.intake..g.day.reference.man."
[21] "Processed.starch.and.sugar.intake..g.day.reference.man"
[22] "Spice.intake..g.day.reference.man."
[23] "Salt.intake..added.to.diet..g.day.reference.man."
[24] "Vegetable.Oil.intake..added.to.diet..g.day.reference.man."
[25] "Animal.Fat.intake..added.to.diet..g.day.reference.man."
[26] "Fruit.intake..g.day.reference.man."
[27] "Green.vegetable.intake..g.day.reference.man."
[28] "Nuts..intake..g.day.reference.man."
[29] "Wheat..average.daily.consumption..g.day."
[30] "All.Grains.Combined..average.daily.consumption..g.day.reference.man."
[31] "Legume.quantity..average.daily.intake..g.day.reference.man."
[32] "Fish.intake..g.day.reference.man"
[33] "Animal.Food.intake..g.day.reference.man."
[34] "Vitamin.C..average.plasma.level..mg.dL."
[35] "Vitamin.A..retinol..an.animal.form.of.vitamin.A..plasma..ug.dL."
[36] "Beta.Carotene..precursor.to.vitamin.A..plasma..ug.dL."
[37] "Cholesterol..HDL..High.density.lipoprotein..cholesterol..plasma..mg.dL."
[38] "Cholesterol..total.cholesterol..plasma..mg.dL."
[39] "First.Pregnancy.Age..average.over.all.mothers"
[40] "Antibiotic.Use..percentage.of.people.who.often.use.antibiotics"
[41] "H.Pylori.IgG.Antibody..using.cut.off.300."
[42] "Schistosomiasis..percentage.with.history.of.diagnoxis"
[43] "Malaria..percentage.with.history.of.diagnosis"
[44] "Hepatitis..percentage.with.history.of.diagnosis"
[45] "First.Menstruation..average.age.at.time.of..females."
[46] "Diabetes..age.35.69..stand..rate.100.000."
[47] "Congenital.Anomalies..age.0.4..cumulative.rate.1.000.by.age.5."
[48] "Heart.Disease..Ischaemic.heart.disease..age.35.69..stand..rate.100.000."
[49] "Cancer..all.malignant.neoplasms..age.35.69..stand..Rate.1.000."
```

Near the end we have some rates for certain diseases and in the beginning we have some lifestyle variables. The large bulk of the variables concern diet.

In this section we will focus on the following question. At the county level, which dietary variables influence the average serum cholesterol of the inhabitants?

We plan on selecting an appropriate linear model that will use dietary variables to explain the variance in total cholesterol. Our first step is to rename all of the variables.

```
> County=Cdata[,1]
> Region=Cdata[,2]
> Lat=Cdata[,3]
> Inc=Cdata[,4]
> Ind.prod=Cdata[,5]
```

```
>  Dist.city=Cdata[,6]
>  Literacy=Cdata[,7]
>  TV=Cdata[,8]
>  Food.short=Cdata[,9]
>  Alcohol=Cdata[,10]
>  Aflatoxin=Cdata[,11]
>  Smoke=Cdata[,12]
>  Pick.veg=Cdata[,13]
>  Green.veg.freq=Cdata[,14]
>  Legumes.freq=Cdata[,15]
>  Smoked.food=Cdata[,16]
>  EPA=Cdata[,17]
>  DHA=Cdata[,18]
>  Diet.chol=Cdata[,19]
>  Sat.fat=Cdata[,20]
>  Processed=Cdata[,21]
>  Spice=Cdata[,22]
>  Salt=Cdata[,23]
>  Add.voil=Cdata[,24]
>  Add.afat=Cdata[,25]
>  Fruit=Cdata[,26]
>  Green.veg.int=Cdata[,27]
>  Nuts=Cdata[,28]
>  Wheat=Cdata[,29]
>  Grains=Cdata[,30]
>  Legumes.int=Cdata[,31]
>  Fish=Cdata[,32]
>  Animal.food.int=Cdata[,33]
>  Vit.C=Cdata[,34]
>  Vit.A=Cdata[,35]
>  Beta.Car=Cdata[,36]
>  HDL.chol=Cdata[,37]
>  Total.chol=Cdata[,38]
>  Preg.age=Cdata[,39]
>  Antib.use=Cdata[,40]
>  H.pylori=Cdata[,41]
>  Schist=Cdata[,42]
>  Malaria=Cdata[,43]
>  Hep=Cdata[,44]
>  Menst=Cdata[,45]
>  Diab=Cdata[,46]
>  Birth.def=Cdata[,47]
>  HD=Cdata[,48]
>  Canc=Cdata[,49]
```

Entering so many definitions can be tedious. Fortunately, the set of commands we have entered can be saved for future use.

7.6.3 Saving a program for later use

In this short subsection we illustrate how to save a program for later use.

In the previous section we imported the 'Late80sCdata.csv' dataset, naming it 'Cdata' before explicitly labeling each of the variables of interest: 'County=Cdata[,1]', 'Region=Cdata[,2]', 'Lat=Cdata[,3]', etc.

In this section we illustrate a method for packaging this work into a program that can be applied in the future, thus eliminating the need to retype all that code.

Having just finished with the writing of our code we can 'copy and paste' it into a text editor.

```
Cdata=read.csv("Late80sCdata.csv",
colClasses = c("character","character",rep("numeric",47)))

County=Cdata[,1]
Region=Cdata[,2]
Lat=Cdata[,3]
Inc=Cdata[,4]
Ind.prod=Cdata[,5]
Dist.city=Cdata[,6]
Literacy=Cdata[,7]
TV=Cdata[,8]
Food.short=Cdata[,9]
Alcohol=Cdata[,10]
Aflatoxin=Cdata[,11]
Smoke=Cdata[,12]
Pick.veg=Cdata[,13]
Green.veg.freq=Cdata[,14]
Legumes.freq=Cdata[,15]
Smoked.food=Cdata[,16]
EPA=Cdata[,17]
DHA=Cdata[,18]
Diet.chol=Cdata[,19]
Sat.fat=Cdata[,20]
Processed=Cdata[,21]
Spice=Cdata[,22]
Salt=Cdata[,23]
Add.voil=Cdata[,24]
Add.afat=Cdata[,25]
Fruit=Cdata[,26]
Green.veg.int=Cdata[,27]
Nuts=Cdata[,28]
Wheat=Cdata[,29]
Grains=Cdata[,30]
Legumes.int=Cdata[,31]
Fish=Cdata[,32]
Animal.food.int=Cdata[,33]
Vit.C=Cdata[,34]
Vit.A=Cdata[,35]
Beta.Car=Cdata[,36]
```

```
HDL.chol=Cdata[,37]
Total.chol=Cdata[,38]
Preg.age=Cdata[,39]
Antib.use=Cdata[,40]
H.pylori=Cdata[,41]
Schist=Cdata[,42]
Malaria=Cdata[,43]
Hep=Cdata[,44]
Menst=Cdata[,45]
Diab=Cdata[,46]
Birth.def=Cdata[,47]
HD=Cdata[,48]
Canc=Cdata[,49]
```

Note that we ommited unnecessary commands such as 'names(Cdata)'. Also, the prompts are ommited.

The next step is to save the code. We save our code as 'C.variables.R'. We think of it as a program that first imports data into R and then labels the variables of interest.

The program can be run by typing the following:

```
> source("C:\\Documents and Settings\\u0274545\\Desktop\\C.variables.R")
```

An alternative is to simply 'copy and paste' the code back into R.

Regardless, at this point we assume that the variables of interest have been labeled, and we proceed with our analysis.

7.6.4 Using the F statistic in model selection

We have seen how the F statistic theoretically arises when selecting a model. In this subsection we give an example of how one might proceed when modeling 'Total.chol' in terms of the predictor variables from the dataset named 'Cdata'.

We start by creating a matrix of the relevant variables.

```
> M=matrix(c(Total.chol,Pick.veg,Green.veg.freq,
Legumes.freq,Smoked.food,EPA,DHA,Diet.chol,
Sat.fat,Processed,Spice,Salt,Add.voil,Add.afat,
Fruit,Green.veg.int,Nuts,Wheat,Grains,Legumes.int,
Fish,Animal.food.int,Vit.C,Vit.A,Beta.Car),ncol=25)
```

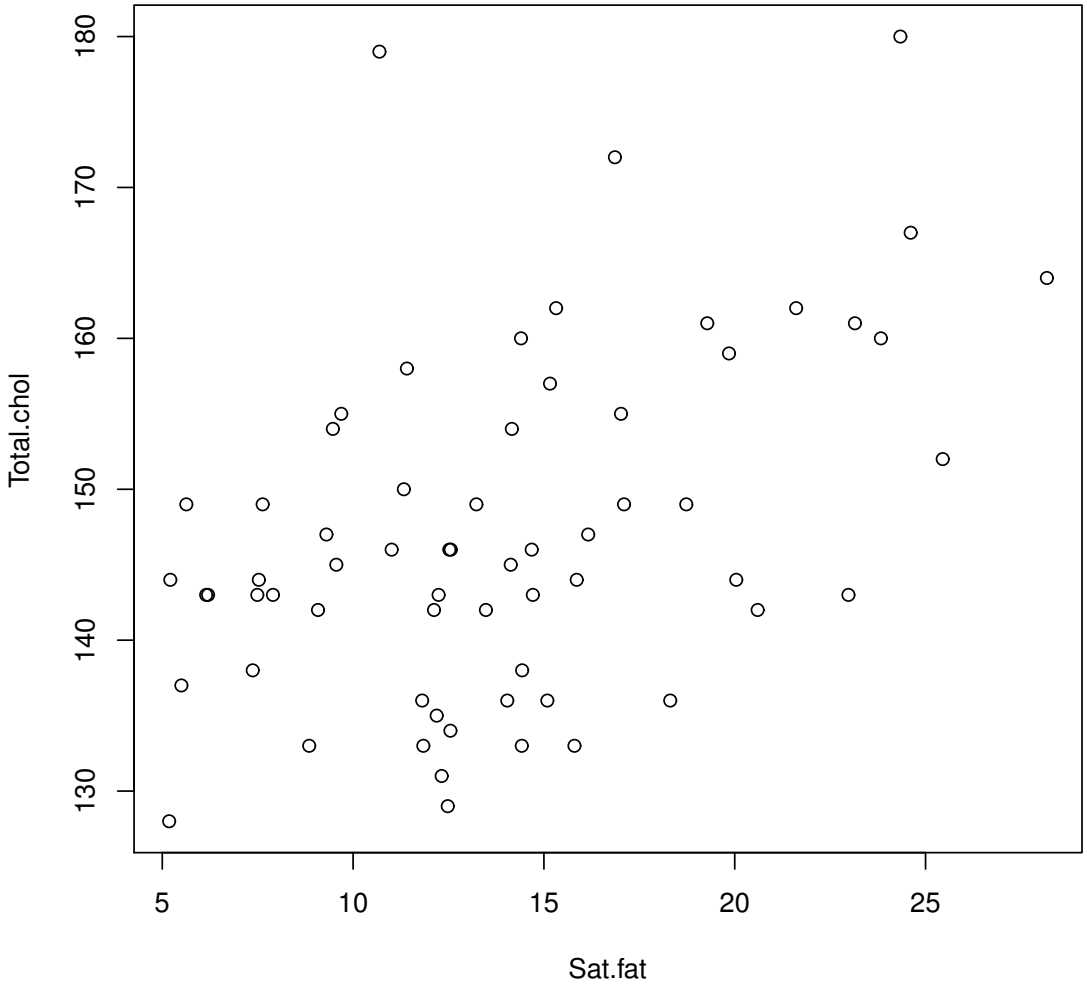
'cor(M)' gives all the bivariate correlations, which could be useful, but we focus first on just the correlations with 'Total.chol'.

```
> cor(M)[,1]
 [1]  1.0000000000 -0.1973348145 -0.1093006699 -0.1620376208 -0.0005138826
 [6]  0.5072277737  0.4124833907  0.6030169144  0.4601336657  0.0667673227
[11] -0.1168517897 -0.3229759141 -0.0071552413  0.1033248987  0.0252134393
[16] -0.2263398704  0.2697635859 -0.0481528194 -0.3403142774 -0.2064373012
[21]  0.4962635342  0.5837464345 -0.1510809739 -0.0130059100  0.1105600286
```

From this we see that none of the explanatory variables are dominantly correlated with Total.chol. The largest correlation coefficient of .603 arises when comparing Sat.fat with Total.chol. A bivariate plot can be obtained via

```
> plot(Sat.fat,Total.chol)
```

and the output is displayed in Figure 7.12. While the positive association is evident, there remains room for more explanatory variables to be added. The question remains: which explanatory variables should be included in the final model?



..

Figure 7.12. Consumption of saturated fat explains approximately 36 percent of the variance in cholesterol levels at the county level

Imagine that the two main candidate-models are nested. The larger model is

```
> LMb=lm(Total.chol~Green.veg.freq+Smoked.food+Sat.fat+
Spice+Salt+Green.veg.int+Nuts+Grains+Legumes.int+Fish+Vit.C)
> summary(LMb)

Call:
lm(formula = Total.chol ~ Green.veg.freq + Smoked.food + Sat.fat +
    Spice + Salt + Green.veg.int + Nuts + Grains + Legumes.int +
    Fish + Vit.C)

Residuals:
     Min       1Q   Median       3Q      Max
-10.0594  -3.6029   0.3234   3.7786  14.5712

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    178.81620    8.39706  21.295  < 2e-16 ***
Green.veg.freq  -0.02943    0.01666  -1.767 0.083137 .
Smoked.food     -0.06252    0.03012  -2.076 0.042849 *
Sat.fat          1.13447    0.18952   5.986 2.02e-07 ***
Spice           -0.20210    0.08690  -2.326 0.023977 *
Salt            -0.73017    0.28307  -2.579 0.012764 *
Green.veg.int   -0.02023    0.01016  -1.990 0.051806 .
Nuts             0.33238    0.11738   2.832 0.006573 **
Grains          -0.02155    0.01054  -2.045 0.045912 *
Legumes.int     -0.11080    0.03912  -2.832 0.006561 **
Fish             0.10473    0.03002   3.489 0.000996 ***
Vit.C          -12.86641    3.49478  -3.682 0.000552 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 6.218 on 52 degrees of freedom
Multiple R-squared: 0.7575,     Adjusted R-squared: 0.7062
F-statistic: 14.77 on 11 and 52 DF,  p-value: 2.111e-12.
```

The smaller model is

```
> LMs=lm(Total.chol~Smoked.food+Sat.fat+Spice+Salt+Green.veg.int
+Nuts+Legumes.int+Fish+Vit.C)
> summary(LMs)

Call:
lm(formula = Total.chol ~ Smoked.food + Sat.fat + Spice + Salt +
    Green.veg.int + Nuts + Legumes.int + Fish + Vit.C)

Residuals:
      Min        1Q    Median        3Q       Max
-10.65504  -4.54423   0.08324   3.62928  16.02694

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   165.271774   5.743234  28.777  < 2e-16 ***
```

```
Smoked.food    -0.081566   0.029367  -2.778 0.007516 **
Sat.fat         1.125045   0.195703   5.749  4.3e-07 ***
Spice          -0.268749   0.084025  -3.198 0.002313 **
Salt           -0.762245   0.292062  -2.610 0.011696 *
Green.veg.int  -0.030874   0.008725  -3.539 0.000836 ***
Nuts            0.371017   0.118517   3.131 0.002815 **
Legumes.int    -0.123795   0.036680  -3.375 0.001373 **
Fish            0.114094   0.028391   4.019 0.000183 ***
Vit.C         -12.933707   3.562850  -3.630 0.000630 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 6.428 on 54 degrees of freedom
Multiple R-squared: 0.7309,     Adjusted R-squared: 0.686
F-statistic: 16.29 on 9 and 54 DF,  p-value: 1.650e-12.
```

This section began with theorem 7.6 which explained how to test whether or not the larger model is significantly better. The test can be implemented within R by feeding the two models to the 'anova' function.

```
> anova(LMb,LMs)
Analysis of Variance Table

Model 1: Total.chol ~ Green.veg.freq + Smoked.food + Sat.fat + Spice +
    Salt + Green.veg.int + Nuts + Grains + Legumes.int + Fish +
    Vit.C
Model 2: Total.chol ~ Smoked.food + Sat.fat + Spice + Salt + Green.veg.int +
    Nuts + Legumes.int + Fish + Vit.C
  Res.Df     RSS Df Sum of Sq    F  Pr(>F)
1     52 2010.23
2     54 2231.35 -2   -221.12 2.86 0.06632 .
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

We focus on the significance of the observed F statistic and find it to be significant only at the .1 level. This tells us that there is not overwhelming reason for using the larger model. Also, since the adjusted R-squared values are comprable, it make sense to go with the smaller conservative model, as a guard against overfitting.

Exercise 7.7. *Load the dataset 'Late80sCdata.csv' into R and fit a linear model of your choice.*

Use as many of the tools that we have thus far developed in this chapter as you can.

Practice saving your work in an editor.

7.7 Bootstrapping

I pulled myself up out of the marsh by my own pigtail

Baron Munchausen

You will never lift yourself by pulling at your own boot-straps

Robert Patterson

[1] The Baron, it seems, has been able to rely on his ability to pull himself up by his own pigtail, and we, so far, have been able to rely on parametric assumptions (think normality) when conducting statistical inference. However, at times such parametric assumptions do not hold. Determined to do statistics nonetheless, we pull ourselves up by our own bootstraps, employing resampling techniques. This section illustrates (within the context of a linear model) the technique known as bootstrapping.

7.7.1 A confidence interval for the mean

We have seen how 'apply' can act on, for example, each column of a matrix, and thus allows us to avoid writing loops (for i in ...).

'lapply' is similar to 'apply'. It operates on a list. For example the following x,

```
> x=list(a=rnorm(10),b=rexp(10),c=rcauchy(10))
> x
$a
 [1] -0.9534184  0.3045003 -0.8845768 -0.6461511 -0.1899758  1.0746331
 [7] -0.5746000 -0.3089100  0.4369594 -0.1930797

$b
 [1] 0.46242547 2.21445077 0.04118299 0.62537203 1.17781528 0.76678423
 [7] 0.24473969 0.22980972 0.35268908 0.81141186

$c
 [1] -0.965140843  2.963351741 -0.385170142 -0.291276007 -0.609883961
 [6] -1.909854065 12.261237094  0.452797580  0.009033186  2.860864940
```

is a list. It is not technically a matrix-type object and thus can not be fed to the 'apply' function. However, we can simultaneously compute the means for each of a,b, and c using the 'lapply' function.

```
> means=lapply(x,mean)
> means
$a
```

[1] 18th century quotes taken from 'World Wide Words', [10]

```
[1] -0.1934619

$b
[1] 0.6926681

$c
[1] 1.438596
```

'lapply' allows us to write a loop. In the above example, the loop can be thought of as follows. For a compute the mean. For b compute the mean. For c compute the mean.

Of course, in this section we will be resampling. We will start with some data.

```
>data=rnorm(100)
```

Our goal is to construct a set of synthetic samples by resampling with replacement from our original sample, which we have named 'data'.

If we want say 500 synthetic samples we can imagine a procedure such as 'for i in 1:500 resample from data'. We would like for this procedure to record each of the 500 synthetic samples. All of this can be accomplished using 'lapply' as follows.

```
>resamp.data=function(i) sample(data,replace=T) ### defining a
resampling function
```

Note that this function differs from functions that we have previously met. Instead of a function of 'x' we have a function of 'i'. It will be fed a sequence, such as '1:50' or '1:500' and is pre-programmed to always do the same thing, namely resample with replacement from 'data'. The sequence determines how many times we should resample.

In the previous scenerio, we used 'lapply' on a list, namely

x=list(a=rnorm(10),b=rexp(10),c=rcauchy(10))

and for each item in the list computed the mean. This time, our list will be a simple sequence, 1:500, and for each item (index) we will compute the prepackaged 'resamp.data' function which resamples our data as described above. This will result in 500 synthetic samples.

```
>synth.samples=lapply(1:500,resamp.data)
```

The result is an unnamed list of the 500 synthetic samples. For example, the 15th synthetic sample is

```
> synth.samples[15]
[[1]]
  [1]  2.07566386 -1.27182058 -0.83461773 -1.02764985 -0.24293589 -1.27091802
  [7]  0.91935221  1.42694438 -1.32802811 -0.06141496  0.59010815  0.37866097
 [13]  0.43709147 -1.17247308 -0.80527787  0.15046183 -0.20726564  1.68081722
 [19] -1.11222215 -1.50474613 -0.04799725 -0.80527787 -0.72021862 -1.50474613
```

```
 [25]  0.44841446  2.28949430 -0.10948591 -0.11091038  1.87521274 -0.33515917
 [31]  0.15046183 -1.45194895 -1.14418815 -1.41769001 -1.14418815 -1.27182058
 [37]  0.84572374 -0.06141496  0.55209169  1.71354850 -1.17247308  0.43158884
 [43] -0.49439165  0.26514391 -1.02764985 -0.83461773  0.43709147 -0.73443032
 [49] -1.49806218 -0.49439165  0.44841446 -0.65035653 -1.08605036 -0.10948591
 [55]  1.25069106 -0.98369239 -0.65035653  1.25069106  0.45471066 -0.65035653
 [61] -1.49806218 -0.72021862 -1.49806218  0.13191304  1.74112239  0.84572374
 [67]  0.44725413  0.36160674 -0.04799725  1.43629372  0.14258220 -0.11091038
 [73] -0.10948591  0.74205804 -0.83461773 -0.70597135 -0.51578094  1.74112239
 [79] -1.17247308  0.36160674  1.43629372  0.44725413 -0.98369239 -1.02764985
 [85]  0.06654356 -0.86121665 -0.94700884 -0.65035653 -0.72545416  1.74112239
 [91] -1.54503369  0.84572374 -1.54503369  0.55209169  0.26514391 -0.70597135
 [97] -1.01766898 -0.51578094 -1.17247308  0.79588115
```

Typing 'synth.samples' would result in the whole list of 500 items, each a synthetic sample, obtained from our data, of 100 resampled data points.

What good are all these synthetic samples? They can be used to shed light on the variability of a statistic. Lets take the mean for example.

```
> mean(data)
[1] 0.03687088
```

This is our sample mean. It is our point estimate for the population mean 'μ'. The distribution for our sample mean is unknown, unless we make parametric assumptions. In this case we have good reason to assume that the data is normal and $\sigma = 1$, because we simulated the data this way. So we unrealistically know that the sample mean of our data is distributed as $N(0, \sigma/\sqrt{100}) = N(0,.1)$. Somewhat more realistically, and entirely practically, we only assume the normality of the data and use the t-statistic to infer.

```
> t.test(data)

        One Sample t-test

data:  data
t = 0.3388, df = 99, p-value = 0.7354
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.1790374  0.2527791
sample estimates:
 mean of x
0.03687088
```

We get a 95% confidence interval for μ. A second argument can specify the degree of confidence.

```
> t.test(data,conf.level=.9)

        One Sample t-test
```

```
data:  data
t = 0.3388, df = 99, p-value = 0.7354
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 -0.1438010  0.2175427
sample estimates:
 mean of x
0.03687088
```

We get a 90% confidence interval for μ. We could have obtained any $1-\alpha$ confidence interval this way via

$$1-\alpha = P[-t^* < \sqrt{n}\frac{\bar{X}-\mu}{S} < t^*] = P[\bar{X} - \frac{St^*}{\sqrt{n}} < \mu < \bar{X} + \frac{St^*}{\sqrt{n}}$$

where t^* depends on α and is computed with a degrees of freedom of $n-1$. We know that

$$\sqrt{n}\frac{\bar{X}-\mu}{S}$$

is distributed as a t, but only because of our normality assumption for X. Even without a normality assumption, other parametric assumptions can lead to confidence intervals in an analogous way (See Chapter 11 of Bain and Engelhardt [4]). It is the parametric assumption that takes us from the sample to the inferred confidence interval. Without such an assumption, we can still proceed by using the nonparametric bootstrap method.

As described above, we sit with just our single observation of $\bar{x}$,

```
> mean(data)
[1] 0.03687088
```

and without any parametric assumptions it is difficult to see how variable our point estimate might be. We have no choice but to pull ourselves up by our own bootstraps.

We use our sample distribution $F_n(x)$ as our best estimate for the population distribution F. We pull from this distribution F_n, 100 observations and compute the mean, and we repeat this procedure 500 times. This results in 500 means which we use to compute the sample mean-of-the-means which we use to estimate the population mean-of-the-means.

We already have the 500 synthetic samples, listed as 'synth.samples'

Another use of 'lapply' allows us to compute the mean for each item in the list

```
> means=lapply(synth.samples,mean)
```

As a technical side note a cleaner printout can be obtained with

```
> means=sapply(synth.samples,mean)
```

By using sapply instead of lapply for this last step we receive a numeric means vector and can make a histogram. We chop off the extremes of this histogram (the sample of sample means) using the quantile function and the result is our bootstrapped confidence interval.

```
> hist(means)
```

```
> q=quantile(means,probs = 1:19/20)
> q
           5%           10%           15%           20%           25%
-0.1517467777 -0.1093456937 -0.0794119940 -0.0602162160 -0.0395769645
          30%           35%           40%           45%           50%
-0.0161594194  0.0004177782  0.0133346883  0.0263110342  0.0418172185
          55%           60%           65%           70%           75%
 0.0551997900  0.0682768829  0.0809659900  0.0955305758  0.1150101837
          80%           85%           90%           95%
 0.1325327592  0.1540761488  0.1802479537  0.2130896762
```

```
> CI=c(min(q),max(q))
```

```
> CI
[1] -0.1517468  0.2130897
```

Compare this to the result obtained using t:

```
> t.test(data)[4]
conf.int
[1] -0.1790374  0.2527791
attr(,"conf.level")
[1] 0.95
```

The parametric interval of approximately $(-.18,.15)$. The bootstrapped interval is tighter. This is due to the fact that our synthetic samples, coming from F_n instead of the true F have less variability.

We introduced the bootstrap in this simulated setting in order to see how it compares to the parametric method. It performed reasonably well. Next we see how it can be put to use in practice.

We will again be using the 'Late80sCdata' data set. Last time we imported the data and labeled the variables. We even saved the program that does that so that we can pick up where we left off. First we set the directory and then run the saved program which leaves us with all variables well defined and labeled as before.

```
> getwd()
[1] "C:/Documents and Settings/u0274545/My Documents"
> setwd("C:/Documents and Settings/u0274545/Desktop")
> source("C:\\Documents and Settings\\u0274545\\Desktop\\C.variables.R")
```

As a reminder. We named the dataset 'Cdata'. A list of variables can be obtained with 'ls()'.

```
> ls()
 [1] "Add.afat"        "Add.voil"        "Aflatoxin"       "Alcohol"
 [5] "Animal.food.int" "Antib.use"       "Beta.Car"        "Birth.def"
 [9] "Canc"            "Cdata"           "County"          "DHA"
[13] "Diab"            "Diet.chol"       "Dist.city"       "EPA"
[17] "Fish"            "Food.short"      "Fruit"           "Grains"
[21] "Green.veg.freq"  "Green.veg.int"   "H.pylori"        "HD"
[25] "HDL.chol"        "Hep"             "Inc"             "Ind.prod"
[29] "Lat"             "Legumes.freq"    "Legumes.int"     "Literacy"
[33] "Malaria"         "Menst"           "Nuts"            "Pick.veg"
[37] "Preg.age"        "Processed"       "Region"          "Salt"
[41] "Sat.fat"         "Schist"          "Smoke"           "Smoked.food"
[45] "Spice"           "Total.chol"      "TV"              "Vit.A"
[49] "Vit.C"           "Wheat"
```

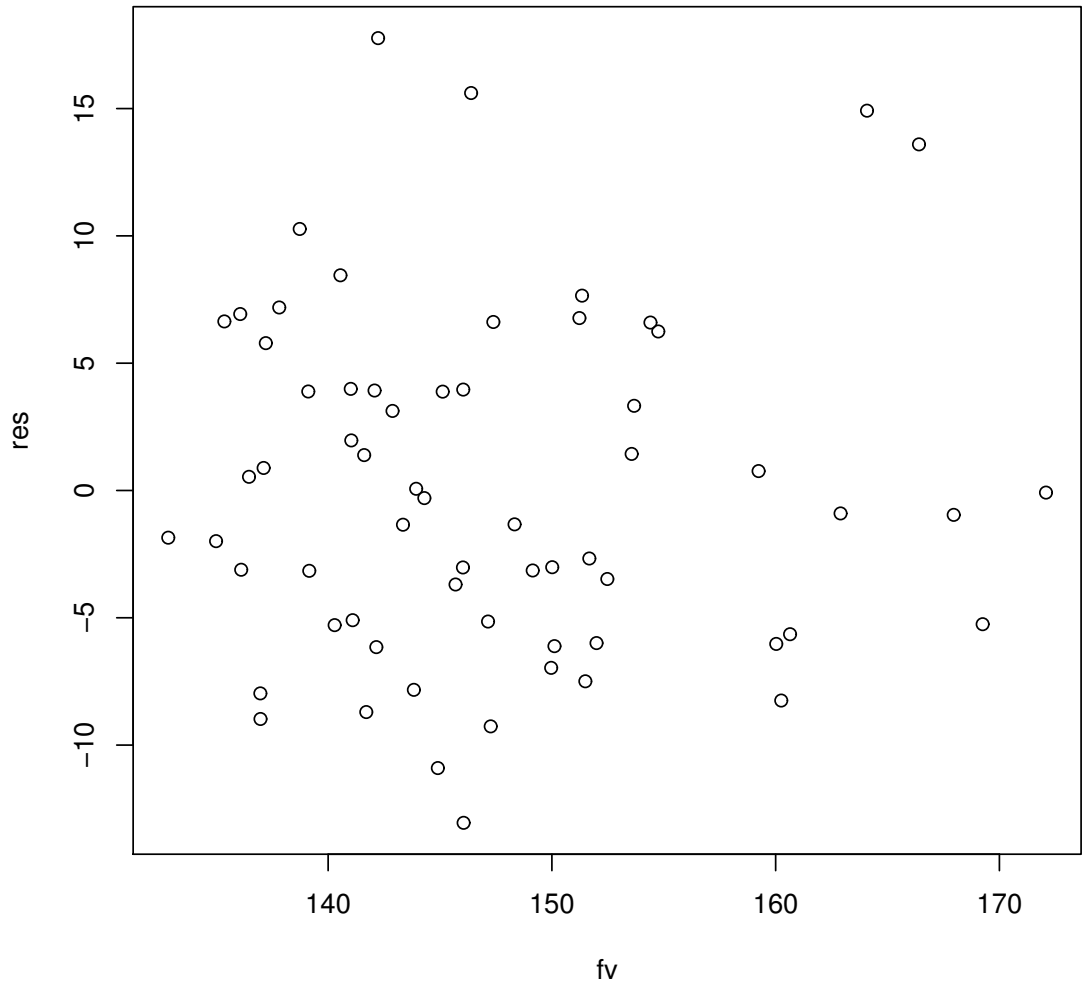
One might reason their way to the following model of Total.chol

```
> LM=lm(Total.chol~Green.veg.freq+Smoked.food+Sat.fat+Salt+Nuts+Legumes.int+Fish)
> summary(LM)

Call:
lm(formula = Total.chol ~ Green.veg.freq + Smoked.food + Sat.fat +
    Salt + Nuts + Legumes.int + Fish)

Residuals:
     Min       1Q   Median       3Q      Max
-13.0540  -5.2612  -0.9298   3.9698  17.7672

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    151.28531    4.55672  33.200  < 2e-16 ***
Green.veg.freq  -0.04691    0.01526  -3.074  0.00326 **
Smoked.food     -0.05908    0.03276  -1.803  0.07669 .
Sat.fat          1.27154    0.21457   5.926 2.00e-07 ***
Salt            -0.91253    0.32073  -2.845  0.00619 **
Nuts             0.41389    0.13086   3.163  0.00252 **
Legumes.int     -0.08507    0.04376  -1.944  0.05693 .
Fish             0.10239    0.03176   3.223  0.00211 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 7.158 on 56 degrees of freedom
Multiple R-squared: 0.654,      Adjusted R-squared: 0.6107
F-statistic: 15.12 on 7 and 56 DF,  p-value: 6.545e-11
```

To check the standard model assumptions we plot the following.

```
> res=LM$residuals
> fv=fitted.values(LM)
> plot(fv,res)
```

and the result is displayed in Figure 7.13. This looks OK. The residuals are not quite normal though.

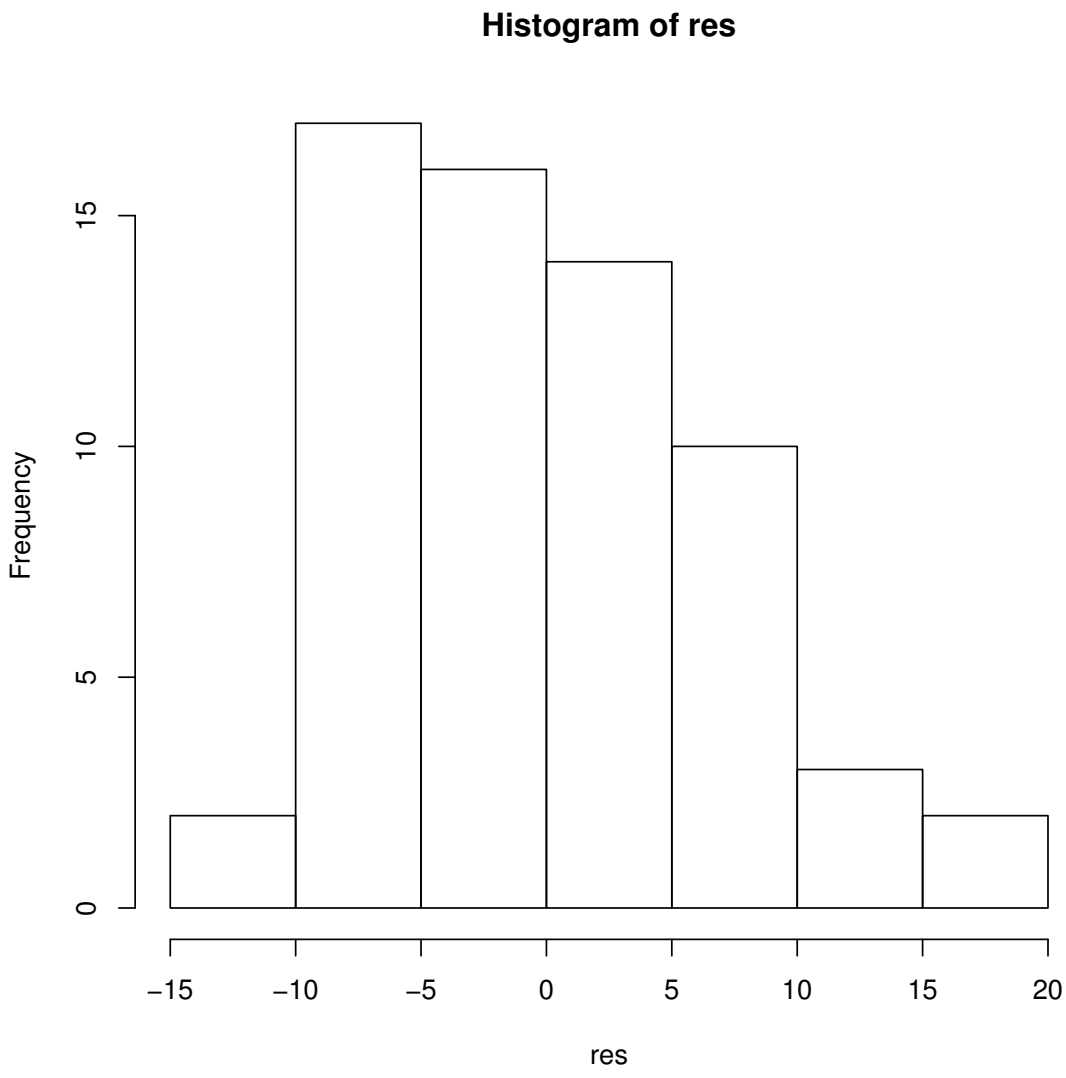


..

Figure 7.13. Residuals plotted against the fitted values for the linear model 'LM'.

```
> hist(res)
```

creates a histogram as displayed in Figure 7.14. This casts a little doubt on the reliability of the printout; after all the printout is computed assuming normality of the errors. How reliable then is our adjusted R-squared value, computed at 0.6107? We can bootstrap the residuals to find out.



..

Figure 7.14. Histogram for the residuals from the linear model 'LM'.

7.7.2 Bootstrapping the residuals

Our aim is to create synthetic y observations and then refit the model for these synthetic y observations, 500 times over. For each synthetic model we will compute an adjusted R2 value and then analyze the sample distribution. The synthetic y observations will be obtained by resampling the residuals.

```
> resamp.res=function(i) sample(res,replace=T) ### defining a resampling
function
> synth.res=lapply(1:500,resamp.res)
```

We now have a list 'synth.res' where each item in the list is a vector of 64 synthetic residuals. The corresponding list of synthetic y values can then be obtained as follows. Note the first few steps simply label the design matrix, X, and the vector of coefficients, betas.

```
> ones=rep(1,64)
> X=matrix(c(ones,Green.veg.freq,Smoked.food,Sat.fat,Salt,
Nuts,Legumes.int,Fish),ncol=8)
> betas=summary(LM)$estimate

> res_to_y=function(x) X%*%betas+x    ### x will be a vector of synthetic residuals
> synth.y=lapply(synth.res,res_to_y)
```

We now have a list of synthetic y vectors 'synth.y'. We use this to obtain a list of synthetic linear models.

```
> y_to_LMs=function(x) lm(x~Green.veg.freq+Smoked.food+
Sat.fat+Salt+Nuts+Legumes.int+Fish)
> synth.LMs=lapply(synth.y,y_to_LMs)
```

We then abstract from this list of synthetic models a list of adjusted R squared values.

```
> LMs_to_adjR2=function(x) summary(x)$adj.r.squared
> synth.adjR2=lapply(synth.LMs,LMs_to_adjR2)
```

Finally, we make the vector numeric so we can perform computations.

```
> synth.adjR2=as.numeric(as.character(synth.adjR2))
```

We could have also used 'sapply' in place of 'lapply' in the definition of synth.adjR2 to obtain a similar result. At this point we can graph a histogram as shown in Figure 7.15.

```
> hist(synth.adjR2)
```

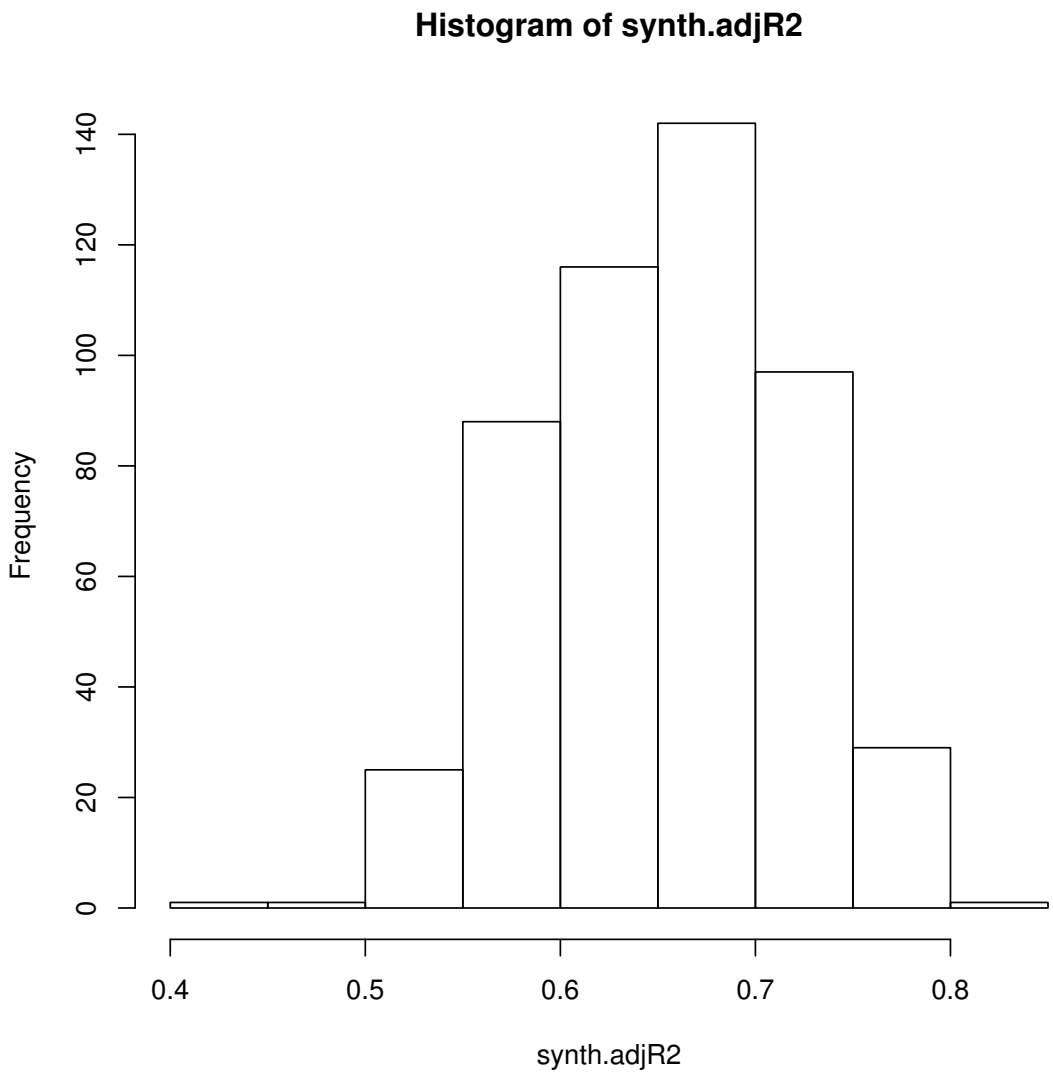
This is interesting indeed. The adjusted R^2 for the original LM, fitted on the real data, printed out as part of the summary as 0.6107. This is not in the center of the bootstrapped histogram. Thus, our confidence interval will not be centered around our observed statistic. The mean of our sample happens to be 0.6548.

```
> mean(synth.adjR2)
[1] 0.6548084
```

The quantiles can then be used to estimate a confidence interval for the population adjusted R^2.

```
> q=quantile(synth.adjR2,probs = 1:19/20)
> CI=c(min(q),max(q))
> CI
[1] 0.5496832 0.7594245
```

This is our bootrapped confidence interval.



..

Figure 7.15. Histogram for the sample of synthetically, bootstrapped adjusted R^2 values

Exercise 7.8. *For the same linear model write code in order to bootstrap the other statistics from the model summary.*

REFERENCES

[1] Mathematica, *Wolfram Mathematica 8,* (http://www.wolfram.com/mathematica/new-in-8/, December 2 2011)

[2] Mathematica, *Wolfram Mathematica 8,* (Retrieved from http://www.wolfram.com/mathematica, Page Created: 2012)

[3] Hewitt and Stromberg (1965). *Real and Abstract Analysis*. New York Heidelberg Berlin: Springer-Verlag.

[4] Bain and Engelhardt (1992). *Introduction to Probability and Mathematical Statistics*. Pacific Grove CA USA: Duxbury.

[5] Norusis, Marija J, *SPSS* Englewood Cliffs, NJ USA: Prentice Hall.

[6] Khoshnevisan, Davar (2007). *Probability*. Providence, Rhode Island: American Mathematical Society.

[7] Moore, David S, *The Basic Practice of Statistics* (United States: WH Freeman and Company, 2010)

[8] Kirk, *Experimental Design: Procedures for the Behavioral Sciences* (Pacific Grove, CA, USA: Brooks/Cole, 1995)

[9] Strang, Gilbert, *Linear Algebra and its Applications* (Australia, Canada, Mexico, Singapore, Spain, United Kingdom, United States: Brooks Cole, 1988)

[10] Quinion, Michael, *World Wide Words - Boot* (Retrieved from http://www.worldwidewords.org/qa/qa-boo2.htm, Page Created: Feb. 2002)

[11] Hornik, Kurt, *R FAQ* (Retrieved from http://cran.r-project.org/doc/FAQ/R-FAQ.html, Page Created: Sept. 2011)

[12] Johnson, Richard; Wichern, Dean, *Applied Multivariate Statistical Analysis - Sixth Edition* (USA: Pearson Prentice Hall, 2007)

[13] Zey, Chelli, *Bartlett's Test* (Retrieved from http://www.itl.nist.gov/div898/handbook/eda/section3/eda357.htm, Page Created: 2003)

[14] Zey, Chelli, *Levene's Test* (http://www.itl.nist.gov/div898/handbook/eda/section3/eda35a.htm, Page Created: 2003)

[15] DasGupta, Anirban, *Asymptotic Theory of Statistics and Probability* (Springer, 2008)

Supplementary material for PhD dissertation University of Utah, Department of Mathematics

Brian Knaeble

August 2012

```
In[1]:= MomentGeneratingFunction[PoissonDistribution[μ], t]

Out[1]= e^((-1+e^t) μ)
```

Figure 1: View of the notebook after computing the moment generating function for Poisson distributions

```
In[1]:= f := (1 / θ Exp[-x / θ]) Boole[0 < x]

In[2]:= exp[θ] = ProbabilityDistribution[f, {x, 0, θ}]

Out[2]= ProbabilityDistribution[((e^(-x/θ) Boole[0 < x]) / θ, {x, 0, θ}]

In[3]:= MomentGeneratingFunction[exp[θ], t]

Out[3]= 1 / (1 - t θ)
```

Figure 2: Applying 'MomentGeneratingFunction' to exponential distributions

```
In[1]:= f := (1 / θ Exp[-x / θ]) Boole[0 < x]

In[2]:= LaplaceTransform[f, x, -t]

Out[2]= -1 / (-1 + t θ)

In[3]:= Simplify[%]

Out[3]= 1 / (1 - t θ)
```

Figure 3: Applying 'LaplaceTransform' to exponential distributions in order to compute moment generating functions

```
In[1]:= f := 1 / (2 θ) Exp[-Abs[x - η] / θ]
        g := 1 / (2 θ) Exp[-Abs[-x - η] / θ]

In[3]:= Assuming[Element[η, Reals], LaplaceTransform[g, x, t] + LaplaceTransform[f, x, -t]]

Out[3]= {-(e^(η/θ) θ) / (-1 + t θ)                                  η ≤ 0
         {-(e^(-η/θ) θ (-1 + 2 e^(t η + η/θ) + t θ)) / ((-1 + t θ) (1 + t θ))   True
        / (2 θ) +

        {(e^(t η) θ) / (1 + t θ)                                    η == 0
        {(e^(-η/θ) θ) / (1 + t θ)                                   η > 0
        {(θ (-2 e^(t η) + e^(η/θ) + e^(η/θ) t θ)) / ((-1 + t θ) (1 + t θ))   True
        / (2 θ)

In[4]:= Simplify[%]

Out[4]= {e^(t η) / (1 - t^2 θ^2)                                    η ≠ 0
        {(e^(-η/θ) (-1 + t θ - e^(2η/θ) (1 + t θ))) / (2 (-1 + t^2 θ^2))   True
```

Figure 4: Applying 'LaplaceTransform' to double-exponential distributions

```
In[1]:= f := (5 / (2 θ^5) x^4) Boole[-θ < x < θ]
        g := (5 / (2 θ^5) x^4) Boole[-θ < x < θ]

In[3]:= dist = ProbabilityDistribution[f, {x, -Infinity, Infinity}]

Out[3]= ProbabilityDistribution[(5 ẋ^4 Boole[-θ < ẋ < θ]) / (2 θ^5), {ẋ, -∞, ∞}]

In[4]:= MomentGeneratingFunction[dist, t]

Out[4]= { (5 (-24 t θ Cosh[t θ] - 4 t^3 θ^3 Cosh[t θ] +                    θ > 0
          24 Sinh[t θ] + 12 t^2 θ^2 Sinh[t θ] + t^4 θ^4 Sinh[t θ])) / (t^5 θ^5)
        { 0                                                                True

In[5]:= Simplify[LaplaceTransform[g, x, t] + LaplaceTransform[f, x, -t]]

Out[5]= { (5 e^(-t θ) (-24 - 24 t θ - 12 t^2 θ^2 - 4 t^3 θ^3 - t^4 θ^4 + e^(2 t θ) (24 - 24 t θ + 12 t^2 θ^2 - 4 t^3 θ^3 + t^4 θ^4))) /   θ > 0
          (2 t^5 θ^5)
        { 0                                                                                                                          True

In[6]:= Simplify[
          LaplaceTransform[g, x, t] + LaplaceTransform[f, x, -t] - MomentGeneratingFunction[dist, t]]

Out[6]= 0
```

Figure 5: LaplaceTransform and MomentGeneratingFunction give equivalent results

```
In[1]:= F := Piecewise[{{x / 4 + 1 / 2, -1 ≤ x ≤ 1}}, ArcTan[x] / π + 1 / 2]

In[2]:= dist2 = ProbabilityDistribution[{"CDF", F}, {x, -Infinity, Infinity}]

Out[2]= ProbabilityDistribution[{ 1/4                  -1 ≤ ẋ ≤ 1
                                { 1 / ((1 + ẋ^2) π)    True      , {ẋ, -∞, ∞}]

In[3]:= MomentGeneratingFunction[dist2, t]

Out[3]= 1 / (4 π t) e^(-i t) (π t + e^(2 i t) π t + 2 i t Gamma[0, (-1 - i) t] - 2 i e^(2 i t) t Gamma[0, (-1 + i) t] -
          2 i t Gamma[0, (1 - i) t] + 2 i e^(2 i t) t Gamma[0, (1 + i) t] - 2 i t Log[-t] +
          2 i e^(2 i t) t Log[-t] + 2 i t Log[(-1 - i) t] - 2 i e^(2 i t) t Log[(-1 + i) t] + 2 i t Log[t] -
          2 i e^(2 i t) t Log[t] - 2 i t Log[(1 - i) t] + 2 i e^(2 i t) t Log[(1 + i) t] + 2 e^(i t) π Sinh[t])
```

Figure 6: Computing the moment generating function for dist2 using 'MomentGenerating-Function'

```
In[4]:= F := Piecewise[{{0, x < -1}, {(x + 1) / 4, -1 ≤ x < 0}, {(x^4 + 1) / 2, 0 ≤ x < 1}, {1, x ≥ 1}}]
        G := Piecewise[{{0, -x < -1}, {(-x + 1) / 4, -1 ≤ -x < 0}, {(x^4 + 1) / 2, 0 ≤ -x < 1}, {1, -x ≥ 1}}]

In[6]:= -t LaplaceTransform[G, x, t] + t LaplaceTransform[1 - F, x, -t] + 1

Out[6]= 1 - (e^(-t) (1 - e^t + e^t t)) / (4 t) + t (-1/t + (24 - 24 e^t + 24 e^t t - 12 e^t t^2 + 4 e^t t^3 + t^4) / (2 t^5))
```

Figure 7: Computing the moment generating function via the most-general Laplace transform formula

```
In[1]:= F := Piecewise[{{0, x < -1}, {(x + 1) / 4, -1 ≤ x < 0}, {(x^4 + 1) / 2, 0 ≤ x < 1}, {1, x ≥ 1}}]
        G := Piecewise[{{0, -x < -1}, {(-x + 1) / 4, -1 ≤ -x < 0}, {(x^4 + 1) / 2, 0 ≤ -x < 1}, {1, -x ≥ 1}}]

In[3]:= -t LaplaceTransform[G, x, t] + t LaplaceTransform[1 - F, x, -t] + 1

Out[3]= 1 - (ℯ^-t (1 - ℯ^t + ℯ^t t)) / (4 t) + t (-1 / t + (24 - 24 ℯ^t + 24 ℯ^t t - 12 ℯ^t t^2 + 4 ℯ^t t^3 + t^4) / (2 t^5))

In[4]:= Simplify[%]

Out[4]= (ℯ^-t (-t^3 + 8 ℯ^2 t (-6 + 6 t - 3 t^2 + t^3) + ℯ^t (48 + t^3 + t^4))) / (4 t^4)
```

Figure 8: Using the general Laplace-transform formula on a complicated distribution function

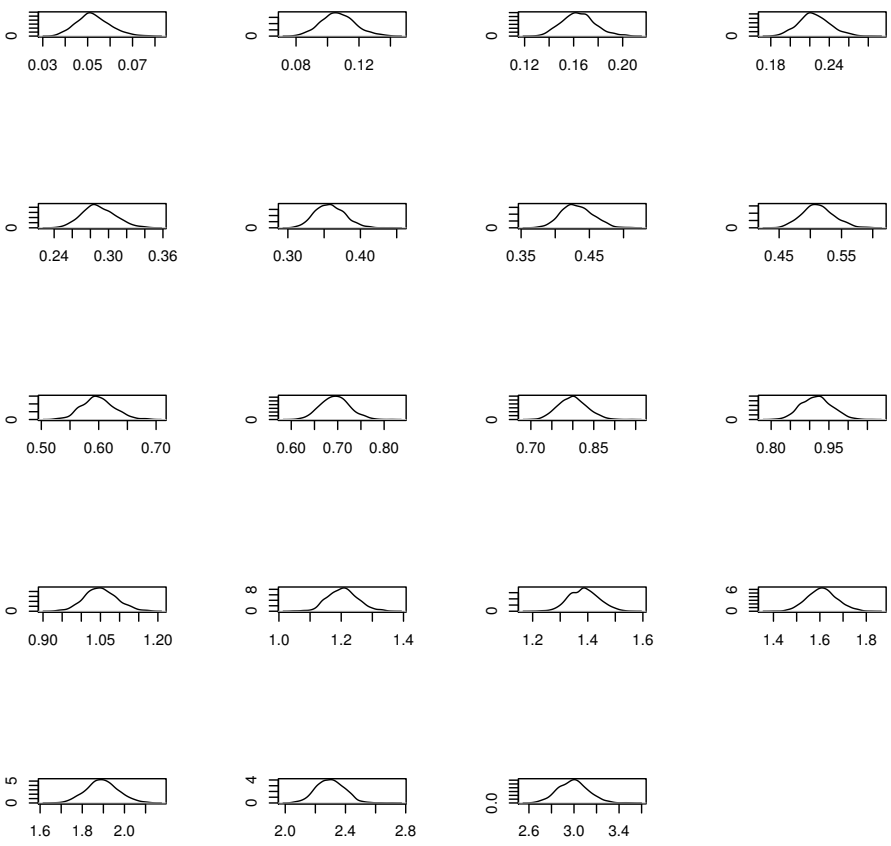


Figure 9: 19 kernel density estimates (standard normal kernal) for the density of various percentiles, 5%(upper left),10%(to the right),...,95%(lower right), of 1,000 standard exponential random variables

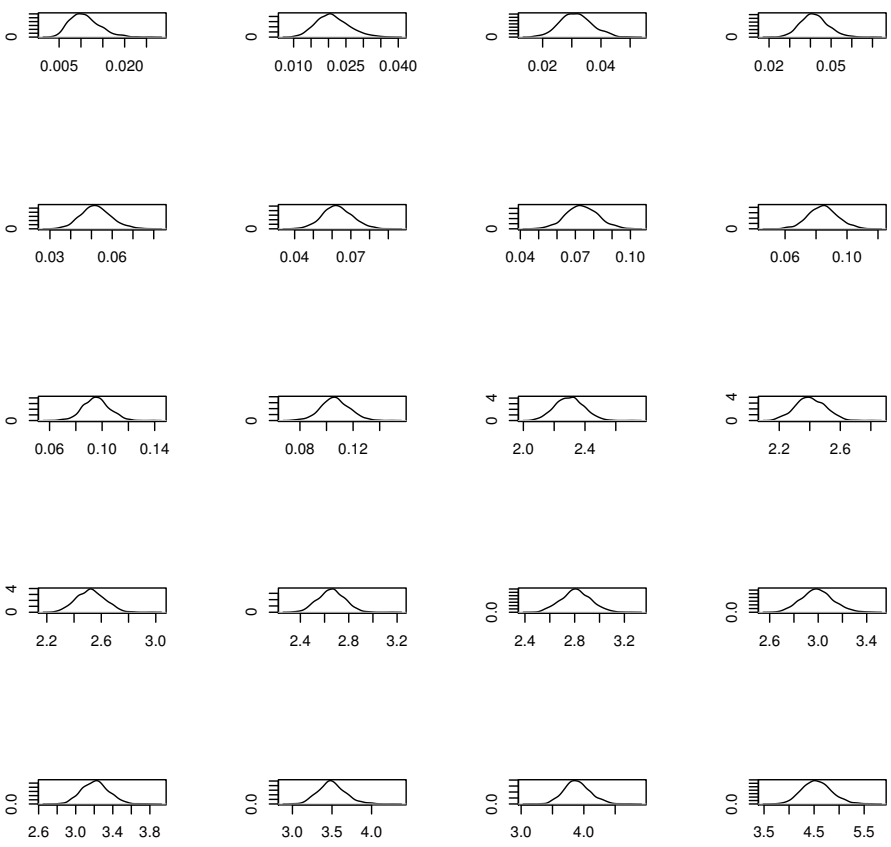


Figure 10: 20 kernel density estimates (standard normal kernal) for the density of various percentiles, 1%(upper left),2%(to the right),...,10%,90%,91%,...,99%(lower right), of $1,000$ standard exponential random variables

Figure 11: 20 kernel density estimates (standard normal kernal) for the density of various percentiles, 1%(upper left),2%(to the right),...,10%,90%,91%,...,99%(lower right), of $1,000$ standard uniform random variables

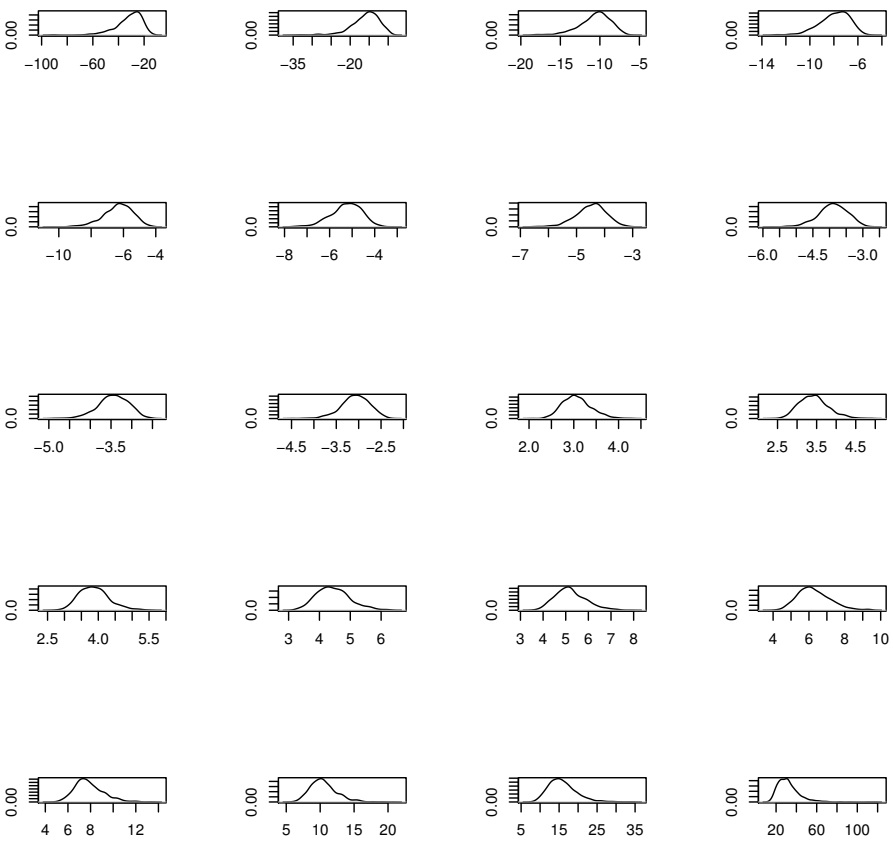


Figure 12: 20 kernel density estimates (standard normal kernal) for the density of various percentiles, 1%(upper left),2%(to the right),...,10%,90%,91%,...,99%(lower right), of $1,000$ standard Cauchy random variables

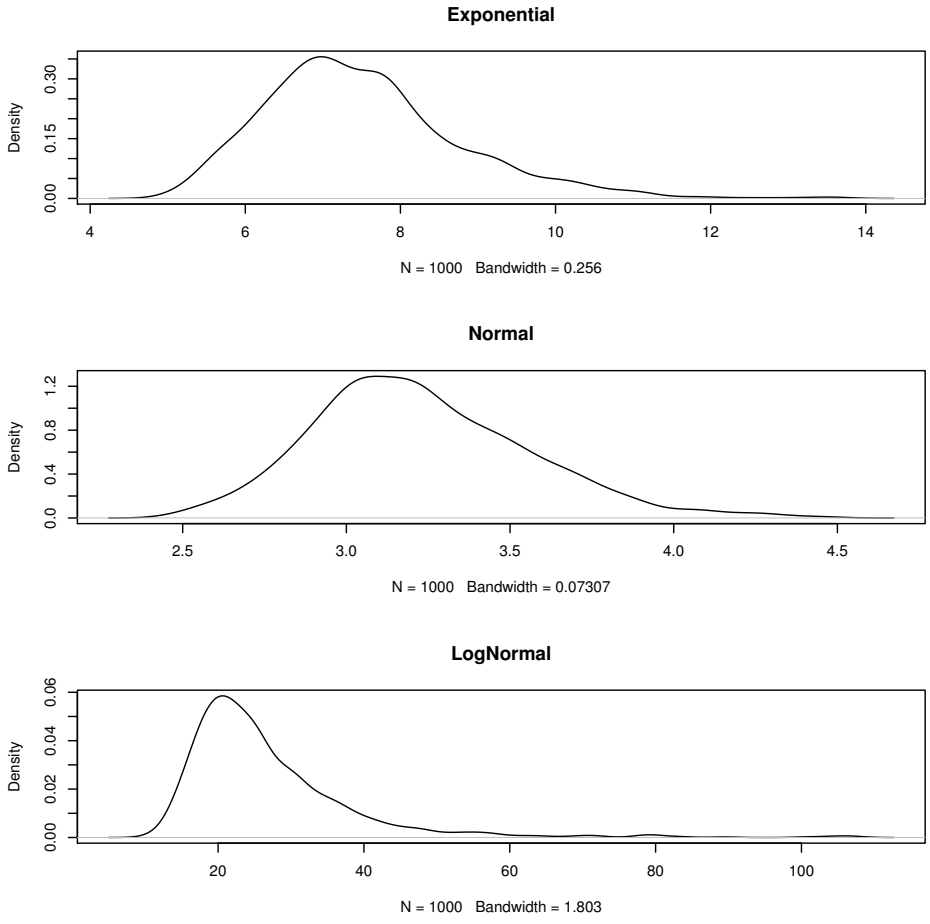


Figure 13: Simulated densities ($n = 1,000$) for $X_{1000:1000}$ where $X \sim$ exponential, normal, log-normal, illustrating Type 1 limiting distributions for the maximal order statistics

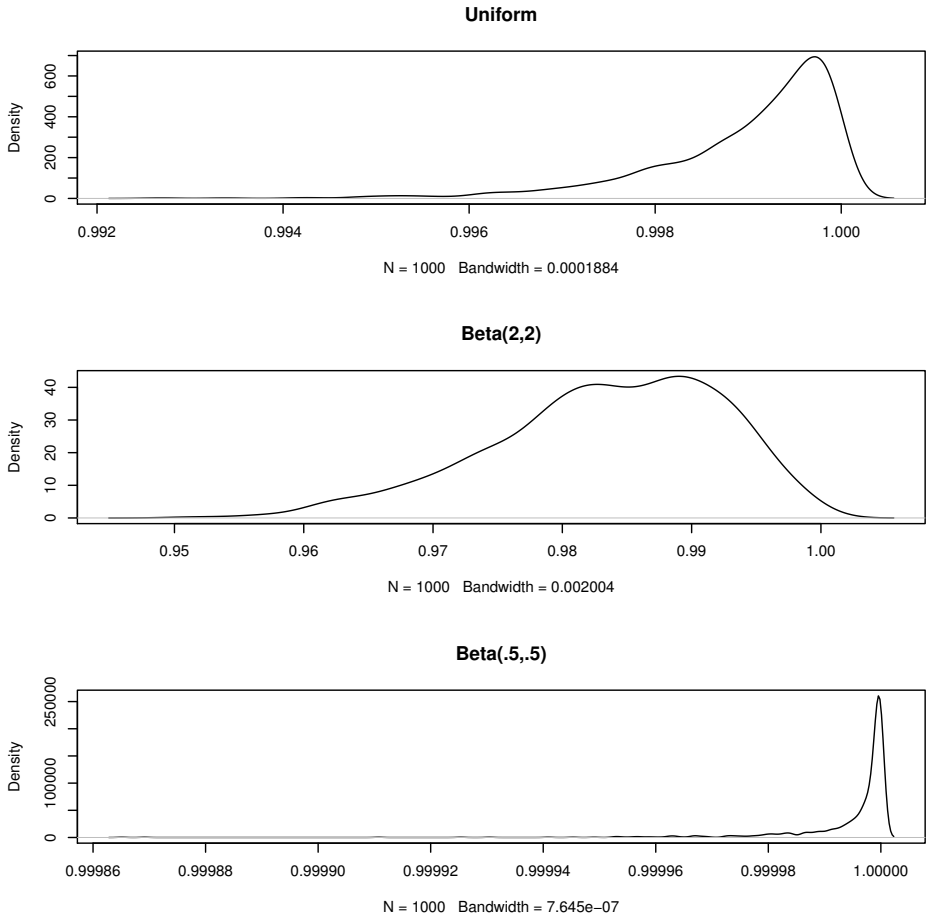


Figure 14: Simulated densities ($n = 1,000$) for $X_{1000:1000}$ where $X \sim$ uniform(0,1), beta(2,2), beta(.5,.5), illustrating Type 2 limiting distributions for the maximal order statistics

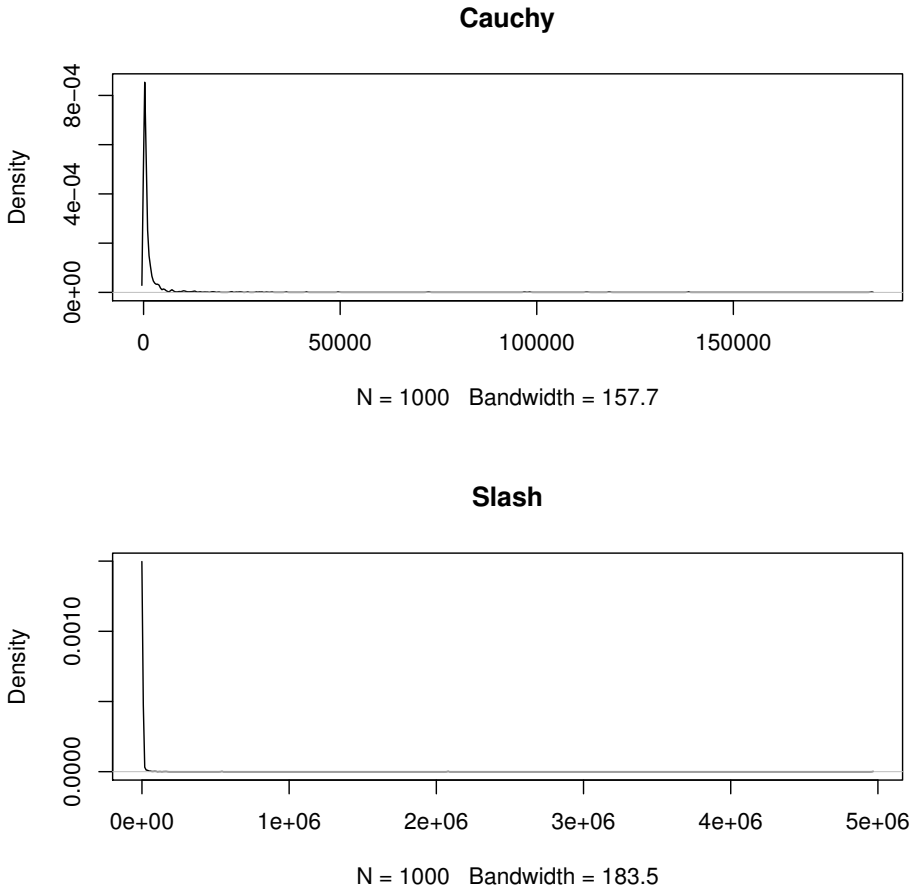


Figure 15: Simulated densities ($n = 1,000$) for $X_{1000:1000}$ where $X \sim$ Cauchy, Slash, illustrating Type 3 limiting distributions for the maximal order statistics

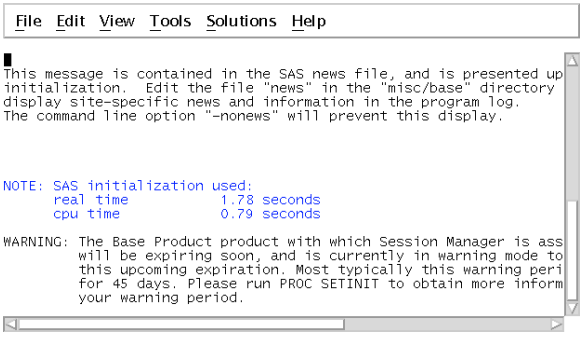


Figure 16: SAS: Log window

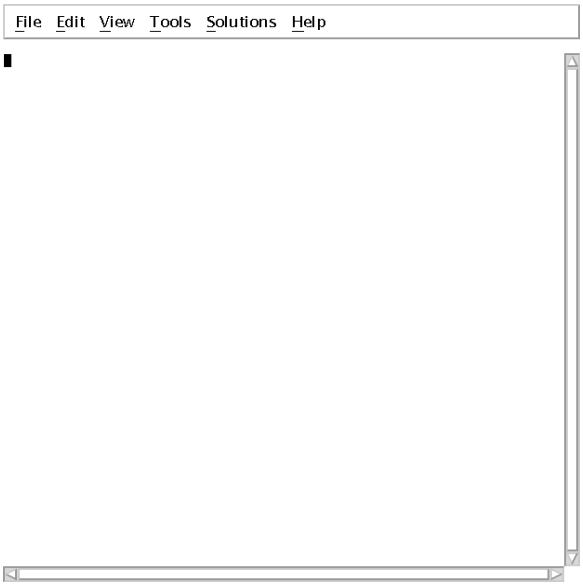


Figure 17: SAS: Output window

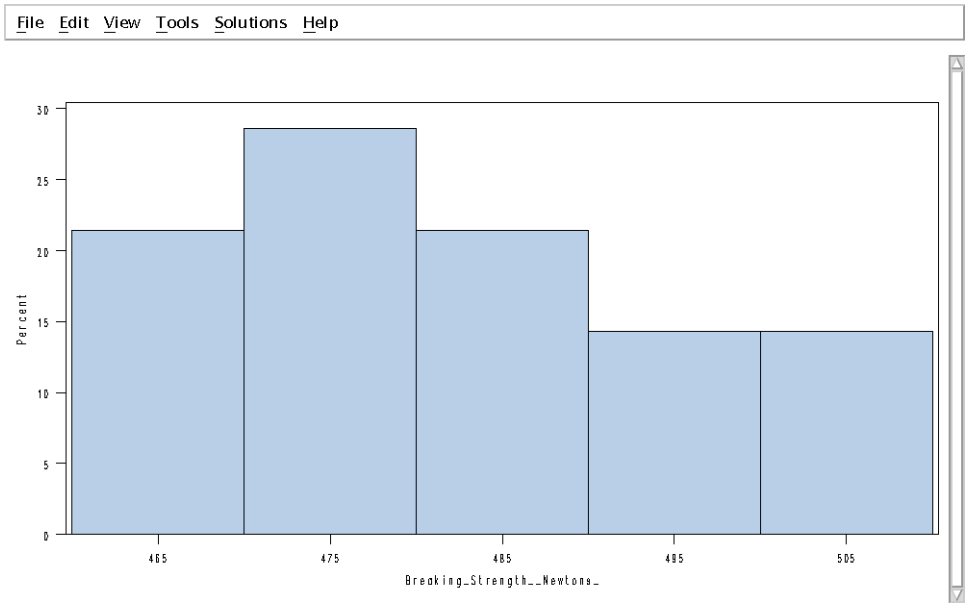


Figure 18: Histogram showing the distribution of the Hockey Stick data